



**EUROPEAN COMMISSION**  
DIRECTORATE-GENERAL  
**Joint Research Centre**

**RTM-IDL 1.0**  
**Unified IDL routines to define a coupled**  
**water-atmosphere system and drive**  
**Radiative Transfer Models**  
**(FEM, Hydrolight)**

MARCO CLERICI

**2005**

**EUR 21556 EN**





**EUROPEAN COMMISSION**  
DIRECTORATE-GENERAL  
**Joint Research Centre**



**RTM-IDL 1.0**  
**Unified IDL routines to define a coupled**  
**water-atmosphere system and drive**  
**Radiative Transfer Models**  
**(FEM, Hydrolight)**

MARCO CLERICI

*European Commission – Joint Research Centre*  
*Institute for Environment and Sustainability*  
*Inland & Marine Waters Unit*  
*TP 272, I-21020 Ispra (VA) – Italy*

## **LEGAL NOTICE**

Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of the following information.

A great deal of additional information on the European Union is available on the Internet. It can be accessed through the Europa server (<http://europa.eu.int>)

EUR 21556 EN

© European Communities, 2005

Reproduction is authorised provided the source is acknowledged

*Printed in Italy*

# Abstract

A set of IDL routines to define a coupled water-atmosphere system and to drive Radiative Transfer simulations is described. This application takes advantage of IDL 6.0 Object Oriented Programming features in order to allow unified description of the physical system, independent of the code to drive, and to ease the maintenance of and the adaptation to a specific model (FEM, Hydrolight, ...). A common library of routines to analyse the results of the simulations is also implemented.

# Table of Contents

<b>Introduction</b>	<b>3</b>
<b>1 Water-Atmosphere system definition and initialisation</b>	<b>5</b>
1.1 General classes . . . . .	7
1.1.1 Vertical Profile . . . . .	7
1.1.2 Phase Function . . . . .	8
1.1.3 Component . . . . .	9
1.2 Atmosphere . . . . .	10
1.2.1 Atmospheric Components . . . . .	10
1.3 Water . . . . .	12
1.3.1 Water Components . . . . .	12
1.3.2 Water derived classes . . . . .	13
1.3.3 Water absorption models . . . . .	14
1.3.4 Water scattering models . . . . .	16
1.4 Other classes . . . . .	18
1.4.1 Sun . . . . .	18
1.4.2 Location . . . . .	18
1.4.3 Output angular definition . . . . .	19
1.4.4 Wavelengths . . . . .	19
1.4.5 Output layer structure . . . . .	19
1.4.6 Bottom Reflectance . . . . .	20
1.5 System Initialisation . . . . .	21
1.6 List of IDL files . . . . .	27
<b>2 Interface to FEM (Finite Element Method)</b>	<b>28</b>
2.1 System to FEM interface . . . . .	29
2.1.1 Atmosphere Optical Properties . . . . .	29
2.1.2 Water Optical Properties . . . . .	32
2.1.3 Band averaged quantities (for SeaWiFS) . . . . .	33
2.2 Example . . . . .	34
2.2.1 Input description . . . . .	36
2.2.2 Run of the example case . . . . .	39
2.2.3 Output analysis . . . . .	39
2.3 List of IDL files . . . . .	46
2.3.1 IDL environment setting . . . . .	46

<b>3</b>	<b>Interface to Hydrolight 4.1</b>	<b>48</b>
3.1	System to Hydrolight interface . . . . .	49
3.2	Example . . . . .	50
3.2.1	Input description . . . . .	52
3.2.2	Run of the example case . . . . .	54
3.2.3	Output analysis . . . . .	55
3.3	List of IDL files . . . . .	57
3.3.1	IDL environment setting . . . . .	57
	<b>Conclusions</b>	<b>59</b>
	<b>Acknowledgements</b>	<b>60</b>
<b>4</b>	<b>Annexes</b>	<b>61</b>
4.1	List of SYS-IDL routines . . . . .	62
4.2	List of FEM-IDL routines . . . . .	93
4.3	List of Hydrolight-IDL routines . . . . .	131
	<b>References</b>	<b>164</b>

# Introduction

The main idea of the package is to separate the physical description of the water-atmosphere system from the Radiative Transfer (RT) program used to solve the radiative equation. This approach enables the User to describe the physical problem to solve in a general way, before choosing a specific tool, like FEM or Hydrolight. The Object Oriented features of IDL 6.0 are used to define general classes that represent a part of the system (e.g. water) and to derive classes for every specific case to be represented (e.g. water case 1). UML (Unified Modelling Language) notation is adopted throughout this document for the classes and methods description (see UML Notation Guide, v1.1, <ftp://ftp.omg.org/pub/docs/ad/97-08-05.pdf> as a reference document).

In Chapter 1, we describe the water-atmosphere system in terms of its physical components: Sun position and Irradiance values, an atmosphere containing gases and aerosols, a geographical location on the water surface, water and a physical bottom in case of finite-depth water bodies. The set of routines developed for this task is called 'SYS-IDL' and this level is referred to as '*System level*' throughout this document.

In order to use a specific RT model, additional routines are designed and implemented; for the time being two Radiative Transfer models are interfaced through 'FEM-IDL' and 'Hydrolight-IDL' modules. In the following this level is indicated as '*RT level*', regardless of the RT tool actually used.

Chapter 2 describes IDL routines to drive the FEM code (Finite Element Method) over a water-atmosphere system. An example of water Case 1 is also discussed, from system definition to results analysis. Chapter 3 does the same, but for Hydrolight 4.1.

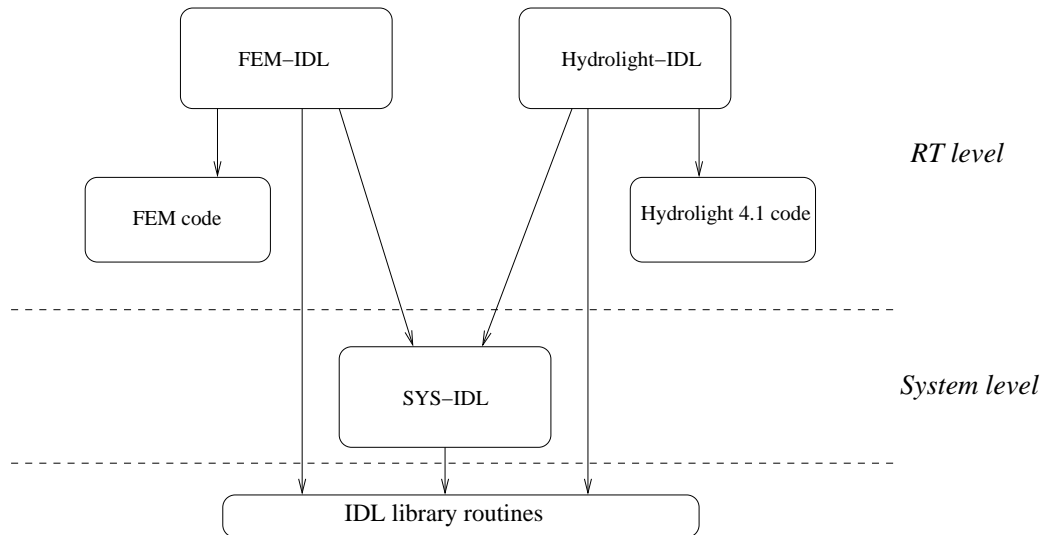


Figure 1: Overall RTM-IDL 1.0 modules hierarchy



The overall hierarchy of the above mentioned modules is displayed in fig. 1: 'FEM-IDL' and 'Hydrolight-IDL' access functionalities of 'SYS-IDL' module and drive the related executables. Furthermore, the three 'IDL' modules use some general purpose procedures defined as 'IDL library routines'.

# Chapter 1

## Water-Atmosphere system definition and initialisation

Fig. (1.1) displays the overall water-atmosphere system and its components: Sun, atmosphere, a geographic location on water surface, water body and physical bottom. For sake of generality no further assumption is done at this level on the nature of the 'objects' represented, everyone of which is modelled by an IDL base class (e.g. 'WAT' for water). From case to case, these 'objects' are specified either by assigning specific properties to them or by defining a class 'derived' from the base one. Throughout this chapter the following notations are adopted:

- bold uppercase style indicates a **CLASS** name.
- italic uppercase is used to denote an *OBJECT* of a given class.
- italic indicates a *member* of a class.

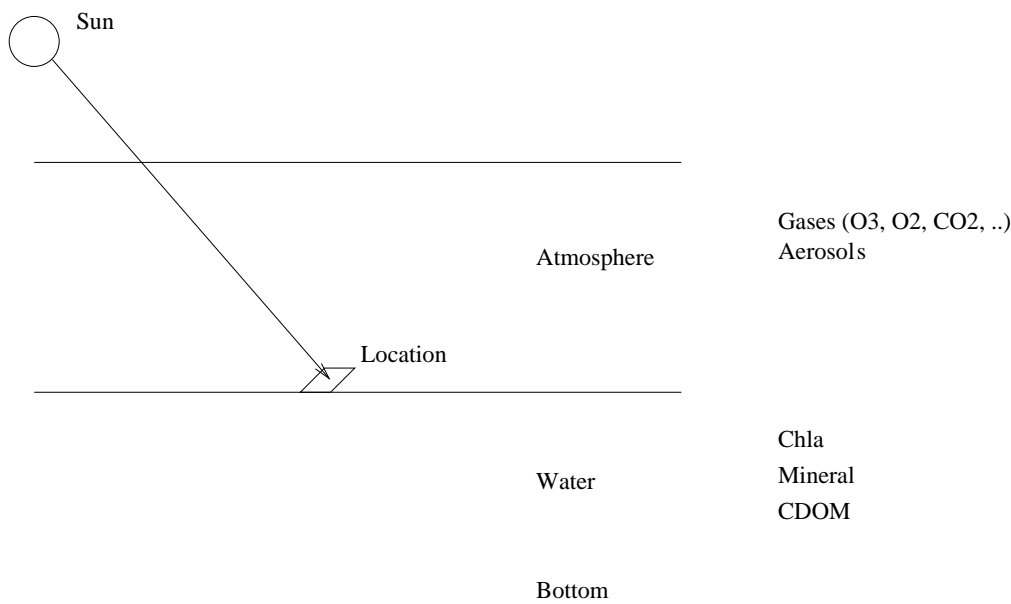


Figure 1.1: Water-Atmosphere system description

The first and main class defined is called **SYSTEM** and aims at defining the whole physical scenario. It contains the objects mentioned above and displayed in fig. (1.2). *id* and *desc* are ASCII strings to identify in concise and more extended way respectively a *SYSTEM* object.

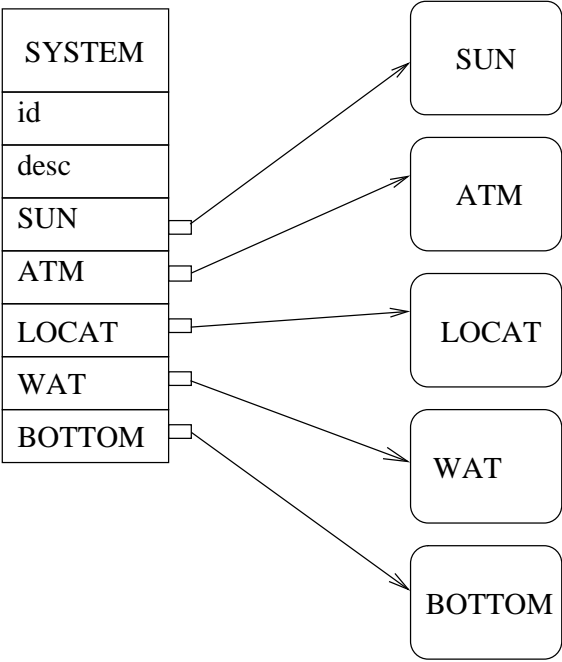


Figure 1.2: System class - 1

Additional classes not referring to the system but to the Radiative Transfer solution are defined as part of **SYSTEM** as they are used by most of the RT models. These additional classes compose figure (1.3): **GRID** contains the viewing zenith and azimuth angles definition, **OLS** (Output Layer Structure) the vertical output layer definition and **WAVE** the list of spectral wavelengths. Please note that **SYSTEM** is displayed in two figures for convenience, but it constitutes an unique class.

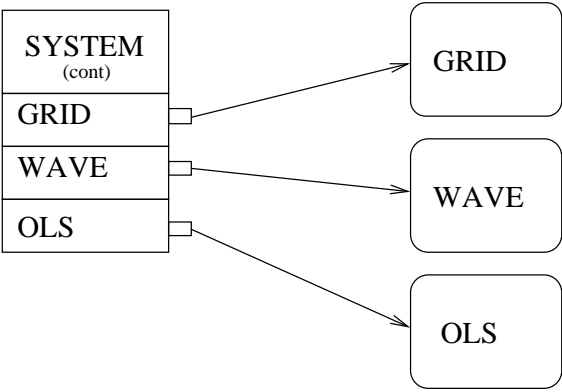


Figure 1.3: System class - 2

## 1.1 General classes

This section contains the so-called 'general purpose' classes, i.e. classes that are used throughout the RTM-IDL 1.0 in many places and can be viewed as a library of objects reusable in IDL applications.

### 1.1.1 Vertical Profile

**VPROFILE** stands for Vertical Profile and defines a scalar quantity as function of altitude, in atmosphere, or depth, in water. It can be used to model the vertical distribution of a gas in atmosphere or the depth-dependent concentration of, e.g., chlorophyll in water.

VPROFILE
X
Y
Status
Filename
Routine

Figure 1.4: VPROFILE class

Members of **VPROFILE** are two float arrays containing the vertical values ( $x$ ) and associated variable values ( $y$ ), plus additional information like object status, name of the file storing the profile distribution and name of the routine producing the values. The two latter are non mandatory as they are used in some **VPROFILE** derived classes but not always.

All classes derived from **VPROFILE** are listed in fig. 1.5.

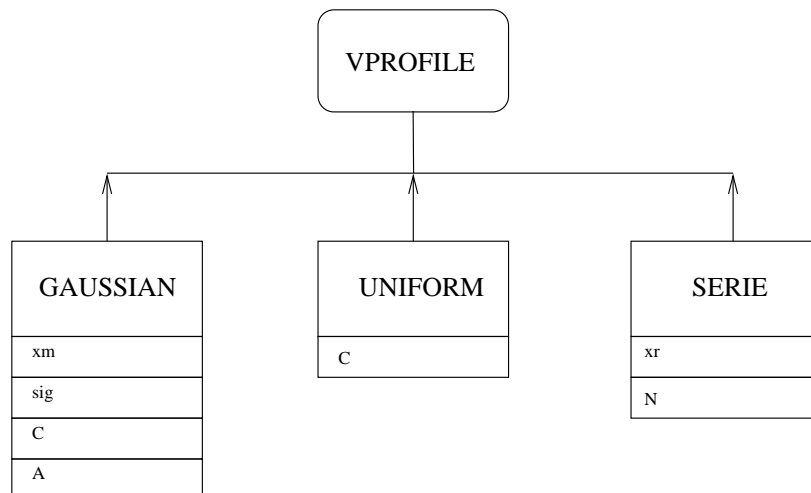


Figure 1.5: VPROFILE derived classes

**GAUSSIAN** is a normally distributed variable, according to the formula:

$$y = C + A \cdot e^{-\frac{(x-x_m)^2}{2 \cdot sig^2}}$$

**UNIFORM** assigns a constant value  $C$  to  $y$ , whatever values is taken by the independent variable  $x$ .

**SERIE** is a generic association of  $x$  and  $y$ -values, with the only restrictions that  $x$  is monotone increasing and the two arrays contain the same number of points. It is intended to be loaded from a text file, whose name is specified by *filename* member of **VPROFILE**. *xr* represents the  $x$ -variable range and  $N$  is the number of points.

### 1.1.2 Phase Function

**PHF** represents a spectral volume scattering phase function, as defined, e.g., in (Mobley 1994, pg. 64). The base class **PHF** contains only an identifier and no normalisation condition is imposed at this level.

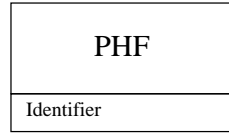


Figure 1.6: Phase Function class

Three phase function types are derived from the base class, as shown in fig. 1.7.

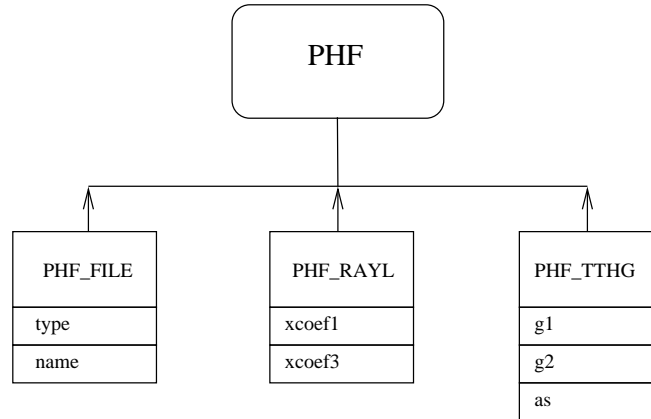


Figure 1.7: Phase Function derived classes

**PHF\_FILE** contains the file type and location. It is an ASCII file listing PHF values versus scattering angles (*type*='Angles') or the associated Legendre coefficients (*type*='Legendre').

**PHF\_RAYL** represents Rayleigh scattering phase function whose Legendre associated coefficients of order 1 and 3 are stored in *xcoef1* and *xcoef3*.

**PHF\_TTHG** is the so-called two-term Henyey-Greenstein phase function as  $\tilde{\beta}$  in Henyey and Greenstein (1941):

$$\tilde{\beta}_{TTHG}(as, g_1, g_2; \psi) = as \cdot \tilde{\beta}_{HG}(g_1; \psi) + (1 - as) \cdot \tilde{\beta}_{HG}(g_2; \psi)$$

and  $g_1$ ,  $g_2$  and  $as$  members are obviously derived from the equation and  $\psi$  is the angle of scattering.

### 1.1.3 Component

**COMP** is a physical component present in the atmosphere or in the water body, like an absorbing gas or a particles distribution in water. It refers to an identified specie, whose properties are modelled in term of absorbing ( $a$ ) and scattering ( $b$ ) coefficients, spectral volume scattering phase function ( $PHF$ ). Its distribution in the medium is described by  $Prof$  that is a **VPROFILE** object as in 1.1.1. Reflecting the generality of the present approach, the class **COMP** contains itself quite a small amount of information and is suitable to be adapted both to marine and atmospheric components. This is done in the following paragraphs by deriving more specific classes from **COMP**.

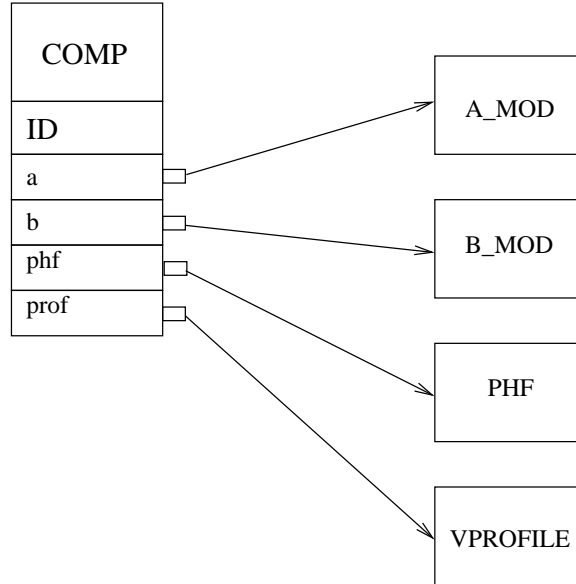


Figure 1.8: Component class

## 1.2 Atmosphere

Atmosphere is defined through the class **ATM** which hosts both physical parameters (e.g. Wind Speed and Surface Pressure) and a list of **COMP** objects (see Section 1.1.3). In the current implementation up to 10 components can be defined, representing different species of absorbing gases or aerosols. No cloud type is provided for the time being.

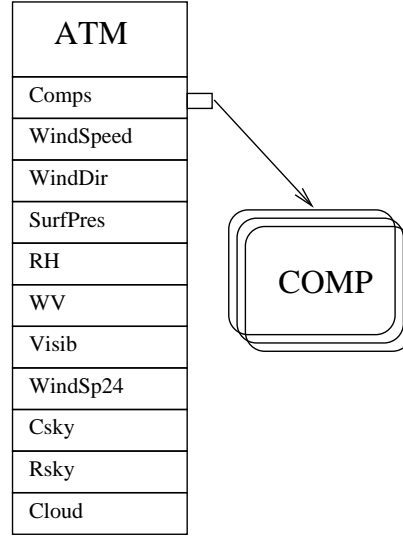


Figure 1.9: Atmosphere class

*WindSpeed*: wind speed at 10 meter altitude in m/s

*WindDir*: direction of wind at 10 meter altitude in degree

*SurfPres*: surface atmospheric pressure in mbar

*RH*: relative humidity in percent

*WV*: total column water vapour content in  $kg/m^2$

*Visibility*: average visibility in km

*WindSp24*: 24 hour average wind speed at 10 meter altitude in m/s

*Csky*: cardioidal parameter as C in Mobley (1994)

*Rsky*: ratio of background-sky to total scalar irradiance

*Cloud*: cloud coverage as in Mobley and Sundman (2000a, pg. 50).

### 1.2.1 Atmospheric Components

**COMP** base class might seem powerful enough to handle whatever kind of component, with no need to 'specialise' it by deriving ad hoc classes for atmosphere or water. Nevertheless this latter approach has been chosen, in order to allow the implementation of specific methods to interface Radiative Transfer models, as described in Chapter 2 and 3.

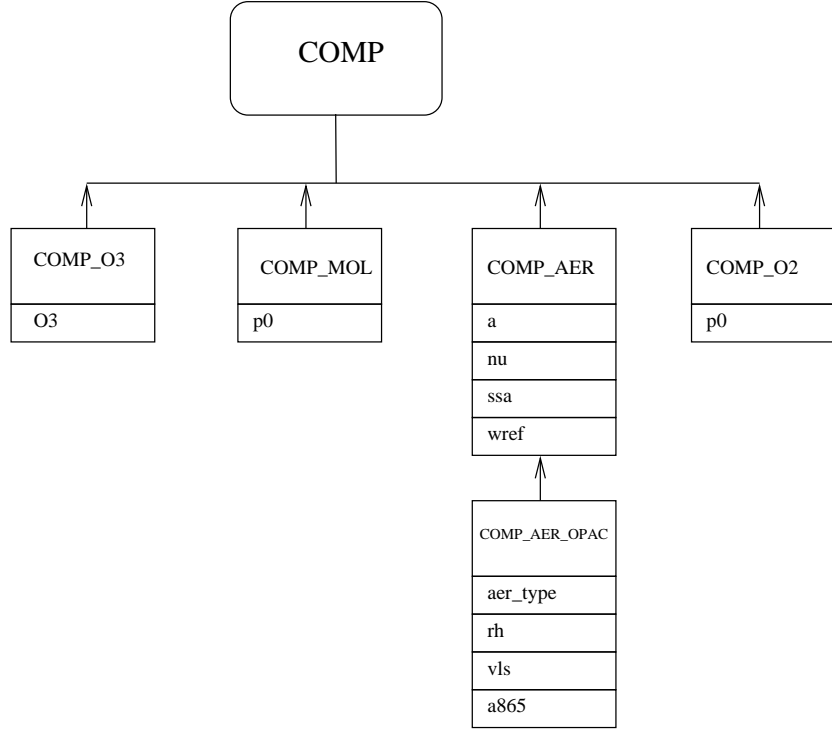


Figure 1.10: Atmospheric components

The derived classes themselves, represented in fig.1.10, add only some fields to the basic **COMP** members: *O3* in **COMP\_O3** is the Ozone total column amount in DBU, *p0* in **COMP\_O2** is the Atmospheric Surface Pressure, intended to be used for the  $O_2$  column rescaling and *p0* in **COMP\_MOL** (molecular component) has the same meaning and purpose.

**COMP\_AER** deals with aerosol particles in the atmosphere: *a* and *nu* are the Ångström coefficient and exponent, referred to the law expressing the spectral dependence of the aerosol optical thickness  $\tau$ :

$$\tau(\lambda) = \tau(\lambda_0) \cdot \left(\frac{\lambda}{\lambda_0}\right)^\nu$$

where  $a = \tau(\lambda_0)$ ,  $nu = \nu$  and  $wref = \lambda_0$  is set by default to 865 nm. Even if this spectral dependency imposes a constraint and is not followed by every aerosol type defined through the RTM-IDL 1.0, these properties have been placed at the top level as this model is defined as the default one. In case of derived classes these **COMP\_AER** members are not used.

**COMP\_AER\_OPAC** refers to OPAC dataset (Optical Properties of Aerosol and Clouds - see Hess et al. (1998)). In particular *aer\_type* corresponds to Table 3 in the mentioned document and *rh* is the relative humidity. Additional member *vls* is the name of the Vertical Layer Structure (VLS, see 2.1) defined to extract the aerosol profile from OPAC dataset and *a865* is the optical thickness at 865. nm used to rescale the aerosol quantity at every wavelength.



## 1.3 Water

**WAT** class represents a finite or infinite-depth water body whose properties are defined by the presence of internal radiation sources (*ISRC*) and up to 10 different components. Bioluminescence, Chlorophyll/CDOM fluorescence and Raman Scattering are considered as water internal sources of radiation (refer to Mobley and Sundman 2000b).

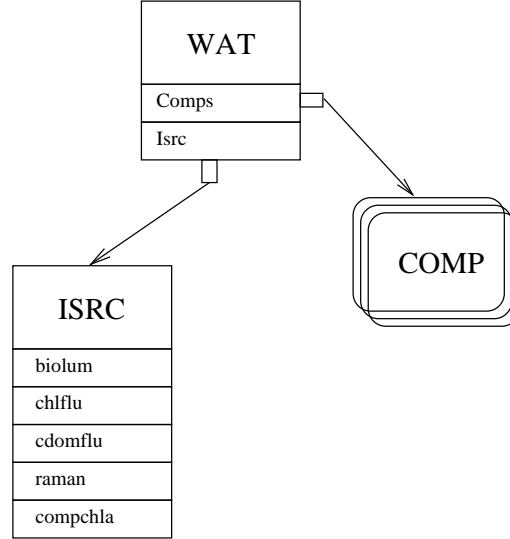


Figure 1.11: Water class

### 1.3.1 Water Components

As done for the atmosphere, water component classes are derived from **COMP**, as shown in fig. 1.12.

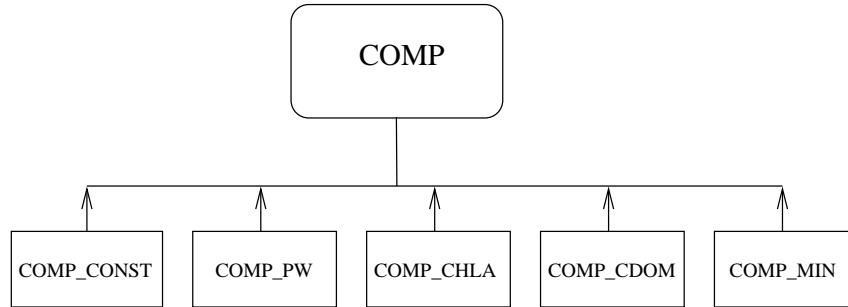


Figure 1.12: Water components

**COMP\_CONST** is a non realistic component whose IOPs do not depend on depth nor wavelength and is fully identified by constant values of absorption/scattering coefficient and phase function. **COMP\_PW** represents the interaction of the radiance with pure water, whose presence is not taken in account from the definition of **WAT** class itself. **COMP\_CHLA** is a generic chlorophyll distribution present in the water body, whose properties are defined by its members, as from fig. 1.8. Of course, different **A\_MOD** and **B\_MOD** objects can be assigned to **COMP\_CHLA**. The

same applies to **COMP\_CDOM** and **COMP\_MIN**, that represent dissolved organic matter and mineral/detritus.

A good question at this point could be why we decided to derive the classes above, if they do not add any property to the base one. The answer is twofold: on one hand, specific methods are defined for the derived classes at '*RT level*'. On the other hand, we preferred to make immediately recognisable water components by this explicit naming in order to ease and make more transparent the initialisation process (see 1.5). For the same reason in the following paragraph derived classes are generated from **WAT**.

### 1.3.2 Water derived classes

Three classes are derived from the base **WAT** one in order to allow the User to easily create and initialise a water object with specific properties.

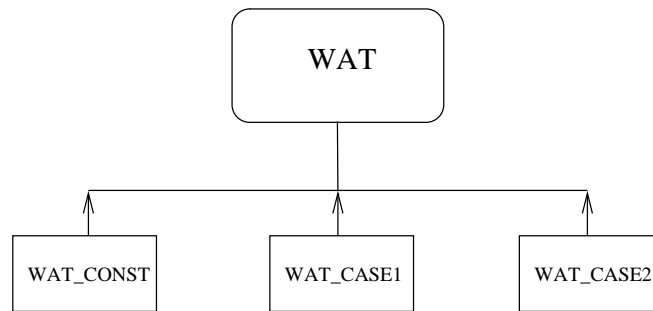


Figure 1.13: Water derived classes

- **WAT\_CONST** has only 1 component of type **COMP\_CONST**
- **WAT\_CASE1** has two components of type **COMP\_PW** and **COMP\_CHLA**
- **WAT\_CASE2** has up to four components of type **COMP\_PW**, **COMP\_CHLA**, **COMP\_CDOM** and **COMP\_MIN**

### 1.3.3 Water absorption models

The responsibility of **A\_MOD** class is to provide the absorption coefficient  $a$  [ $m^{-1}$ ] as function of the wavelength, the component concentration and additional parameters whose values are stored as class members.

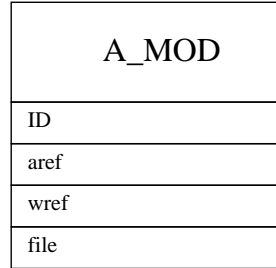


Figure 1.14: Water absorption model

*ID* is a non mandatory string identifier, *aref* the absorption coefficient in  $m^{-1}$  at a reference wavelength *wref* and *file* is the name of an optional file containing absorption coefficient as function of the wavelength. The syntax of this file is not detailed here as it depends on the RT tool used.

Several absorption models are defined deriving classes from the base **A\_MOD**, as shown in fig. (1.15).

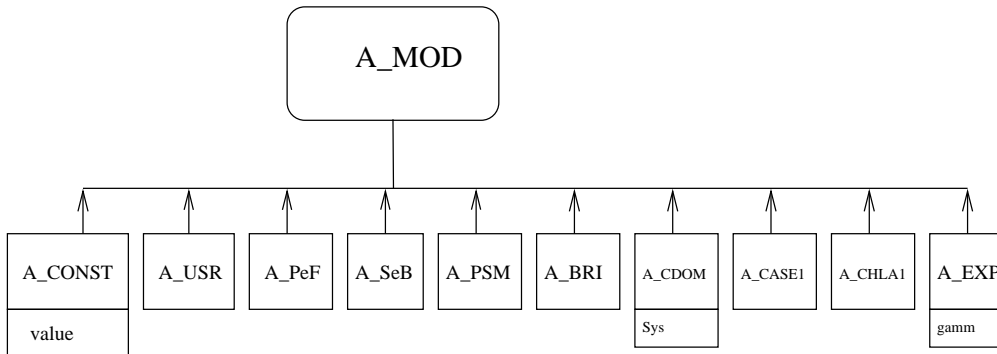


Figure 1.15: Water absorption derived classes

For sake of clarity, class members meaning is explained in Table 1.1, as are the equations providing the absorption coefficient and references to literature.

Class	Members	Equation	Reference
A_CONST	value: a. coeff.	$a(\lambda) = value$	
A_PeF		$a(\lambda)$ tabulated	Pure Water a coeff. (Pope and Fry 1997)
A_SeB		$a(\lambda)$ tabulated	Pure Water a coeff. (Smith and Baker 1981)
A_PSM		$a(\lambda) = 0.06A_{chl}(\lambda)Chla^{0.65}$ <sup>(1)</sup> $A_{chl}$ tabulated <sup>(2)</sup>	<sup>(1)</sup> (Mobley and Sundman 2000b) <sup>(2)</sup> (Prieur and Sathyendranath 1981)
A_EXP	gamma: spectral slope	$a(\lambda) = 0.2a_p(440)e^{-gamma(\lambda-440)}$ $a_p(440)$ is Chla abs. coeff.	(Mobley and Sundman 2000b, pg. 5 )
A_BRI		$a(\lambda) = C_{ph}(\lambda) \cdot Chla^{1-E_{ph}(\lambda)}$ $C_{ph}$ and $E_{ph}$ tabulated	(Bricaud et al. 1995)
A_CDOM	$S_{ys}$ : spectral slope	$a(\lambda) = a(\lambda_0) \cdot e^{-S_{ys}(\lambda-\lambda_0)}$	(Bricaud et al. 1981)
A_CHLA1		$a = a_{Chla} + a_{np}$ where $a_{Chla}$ as in A_BRI $a_{np} = 0.0124 \cdot Chla^{0.744}e^{-0.011 \cdot (\lambda-440)}$	(Bricaud et al. 1995 and 1998)
A_CASE1		$a = a_{Chla} + a_{ys} + a_{np}$ where $a_{Chla}$ as in A_BRI $a_{ys}(440) = a_{Chla}(440) \cdot 0.2$ $a_{ys} = a_{ys}(440) \cdot e^{-0.014 \cdot (\lambda-440)}$ $a_{np} = 0.0124 \cdot Chla^{0.744}e^{-0.011 \cdot (\lambda-440)}$	(Bricaud et al. 1995 and 1998)

Table 1.1: Absorption Coefficient definition

Note that **A\_EXP** is a particular case of **A\_CDOM** model where the reference absorption coefficient is taken from the Chlorophyll one at 440 nm. Therefore it can be used only if a chlorophyll component is defined.

### 1.3.4 Water scattering models

Particle scattering properties are modelled by **B\_MOD** class, whose definition in fig. (1.16) is completely similar to **A\_MOD** class.

B_MOD
ID
bref
wref
file

Figure 1.16: Water scattering model

Scattering models **B\_MOD** are derived from the base class as represented in fig. (1.17). In analysing this picture consider that some of the models use information stored in the base class members.

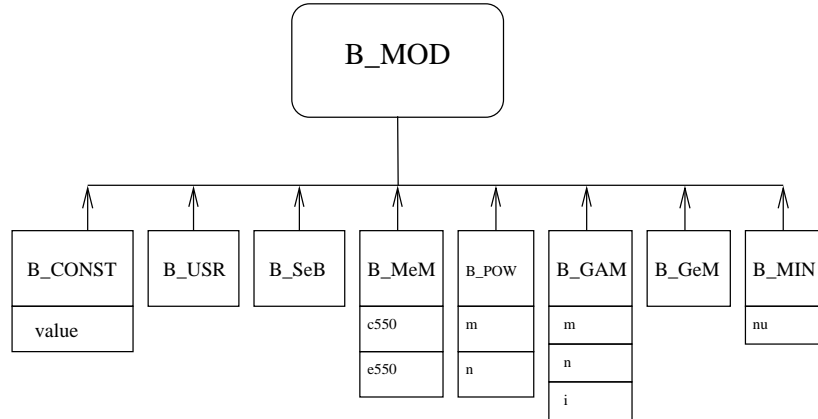


Figure 1.17: Water scattering models

For sake of clarity, class members meaning is explained in Table 1.2, as are the equations providing the scattering coefficient and references to literature.

Class	Members	Equation	Reference
B_CONST	value: b. coeff.	$b(\lambda) = value$	
B_SeB		$b(\lambda)$ <i>tabulated</i>	Pure Water b coeff. (Smith and Baker 1981)
B_MeM	c550: ref. value e550: expon. decay	$b(550) = c550 \cdot Chla^{e550}$ $b(\lambda) = b(550) \cdot (\frac{\lambda}{550})^\nu$ $\nu = -1. \text{ if } Chla < 0.02 \text{ mg m}^{-3}$ $\nu = .5[\log(Chla)0.3] \text{ if } 0.02 < Chla < 2$ $\nu = 0. \text{ if } Chla > 2 \text{ mg m}^{-3}$	(Morel and Maritorena 2001)
B_POW	m: m of power law n: n of power law	$b(z, \lambda) = b_o(\frac{\lambda_o}{\lambda})^m X(z)^n$ $b_o = 0.3$ $X = comp.conc.$	(Mobley and Sundman 2000b, pg. 13)
B_GAM	m: m of GAM law n: n of GAM law i: i of GAM law	$b(z, \lambda) = 0.5(\frac{m\lambda+i}{m\lambda_o+i})X(z)^n$ $\lambda_o = 550nm$	(Mobley and Sundman 2000b, pg. 13)
B_GeM		as B_POW with $m = 1. n = 0.62$	(Mobley and Sundman 2000b, pg. 13) (Gordon and Morel 1983)
B_MIN	nu: exp. decay	$b(\lambda) = b(550) \cdot (\frac{\lambda}{550})^\nu$ $b(550) = comp.conc.$ $nu = \nu$	

Table 1.2: Scattering Coefficient definition

Note that **B\_MIN** model uses the same spectral dependency as in **B\_MeM** but it allows a free definition of the reference scattering coefficient  $b(550)$  and spectral slope  $\nu$ .

## 1.4 Other classes

### 1.4.1 Sun

Sun is simply modelled by its angular position *Zen* and *Azi* and the total irradiance at TOA *Edtot* [ $Wm^{-2}$ ]. Alternative definitions by the universal time and the location are not implemented in the current version of the tool.

SUN
Zen
Azi
Edtot

Figure 1.18: Sun class

### 1.4.2 Location

Location represents a geographical position defined by latitude and longitude, and a given time. Its use is foreseen as an alternative to the SUN definition as in 1.4.1, but is not yet implemented in any RT model used.

LOCAT
Lat
Lon
Time
JDay

Figure 1.19: Locat class

*Lat* is latitude in degree, positive North, *Lon* the longitude, positive East, *Time* UT time expressed as a float (hour and fraction of the hour), *JDay* Julian Day.

### 1.4.3 Output angular definition

Most of RT models have the capability to produce output of directional quantities (i.e. radiance or reflectance) at given viewing angles, that in general can be independent from the internal directional 'discretisation'. **GRID** class allows the setting of these angular values for the RT model output.

GRID
nazi
azi
nzen
equi
zen

Figure 1.20: Grid class

*nazi* is the number of Azimuth angles and *azi* is the list of values. The zenith angle can be defined either by a list of values (*zen*) or setting the total number of angles (*nzen*) and imposing a uniform distribution in the range 0 .. 180 degree through the flag *equi*.

### 1.4.4 Wavelengths

WAVE
nwave
wave

Figure 1.21: Wave class

**WAVE** class defines the wavelength at which RT model computation will occur: *nwave* is the number of wavelengths and *wave* is the list of values in nm.

### 1.4.5 Output layer structure

OLS
Type
ndepth
depths

Figure 1.22: OLS class



RT models produce outputs computed at given 'levels' of the plane-parallel system. These levels can be defined both in terms of geometrical height/depth in atmosphere/water or by providing the related value of optical depth. **OLS** class allows both definitions, with some limitations, in the following way:

- *Type* = 0: levels are defined as *depths* in water from the surface downward, in meter. *ndepth* is the total number of levels.
- *Type* = 1: *depths* is a list of *ndepth* optical thickness values, starting as 0 at Top of atmosphere and increasing downward.

### 1.4.6 Bottom Reflectance

The bottom boundary condition is modelled through the **BOTTOM** object, which can represent both finite and infinite water bodies.

BOTTOM
Type
File
Refl

Figure 1.23: BOTTOM class

*Type* is an integer value used to select the boundary condition type:

- 0 : infinite water body. The IOPs at the deepest depth defined are used to compute the reflectance of the infinitely deep layer of water below the region of interest.
- 1 : irradiance reflectance of the bottom is wavelength independent. A single value is provided through *Refl*.
- 2 : irradiance reflectance of the bottom is wavelength dependent. A filename is provided through *File*.

*File* is the file containing the bottom reflectance as function of the wavelength (if *Type*=2).

*Refl* is the wavelength independent bottom irradiance reflectance (if *Type*=1).

## 1.5 System Initialisation

A *SYSTEM* object is created and initialised through a structure that reflects the hierarchical organisation of the class **SYSTEM** itself. This structure is called **SYS\_IN** and is shown in Table (1.3), where Member is the name of each structure member, Type is its type, column 3 contains values allowed to each member, and column 4 is a typical assignment. Type can be an IDL base type (int, float, string, boolean, pointer to float array, ...), name of a sub-structure, in which case it is indicated in curly braces, a named array indicated within square brackets or a *string for multiple assignment*. The latter is a convenient way to create and initialise a simple object that includes only scalar members (i.e. a single float, integer, string, boolean). An example is the **SUN** object (see fig. 1.18) that is initialised by the following *string for multiple assignment* (called *ma-string* in the following):

[SUN: ZEN=*float* , AZI=*float*, EDTOT=*float*].

A *ma-string* follows an easily recognisable syntax: square brackets as delimiter, name of class, colon, one or more assignments done by the name of the member, equal and value. In case one of the members is not assigned, it takes a default value during initialisation.

A water absorbing model of type **A\_CDOM** can be created by the following *ma-string*:

[A\_CDOM: id='yellsubst', aref=0.2, wref=440., sys=0.014]

Note that it is possible to assign in the same way values to members of the base class **A\_MOD** and of the derived class **A\_CDOM**.

Structure SYS_IN			
Member	Type	Values allowed	Example
id	string	any string	'Test'
desc	string	any string	'System initialisation for Test'
model	string	'FEM', 'HYD'	'FEM'
sun	ma-string	see tab. 1.4	'[SUN : ZEN=0., AZI=0., EDTOT=1.]'
atm	{ATM_IN}	see tab. 1.5	
grid	{GRID_IN}	see tab. 1.7	
locat	ma-string	see tab. 1.8	'[LOCAT : LAT=21.3, LON=0.4, JDAY=131, HH=12.276]'
bottom	ma-string	see tab. 1.9	'[BOTTOM: TYPE=2 ,REFL=.2,FILE="GreenAlgae.txt"]'
wat	{WAT_IN}	see tab. 1.10	
ols	{OLS_IN}	see tab. 1.13	
wave	{WAVE_IN}	see tab. 1.14	

Table 1.3: SYS\_IN initialisation structure

The **SUN** object, part of the system as shown in fig.(1.2), is simple enough to be initialised through a multiple assignments string. Table (1.4) contains details of type and allowed values for each member.

<i>ma-string</i> SUN			
Member	Type	Values allowed	Example
ZEN	float	0. .. 90.	40.
AZI	float	0. .. 180.	50.
EDTOT	float	positive value	78.9

Table 1.4: SUN initialisation string

**ATM** object (see fig. (1.9)) requires a dedicated structure called ATM.IN to be initialised, containing values for the members and string arrays ([ATM\_COMP]) for the atmospheric components. In the current implementation, the order of the components is fixed and is the following: Ozone, Aerosol, molecules for Rayleigh scattering and Oxygen. Therefore there is no need to define the component type during the initialisation phase, and it is enough to specify the members value.

Structure ATM.IN			
Member	Type	Values allowed	Example
WindSpd	float	positive value [m/s]	2.
WindDir	float	0. .. 360.	67.
SurfPres	float	positive value [mbar]	1013.25
RH	float	0. .. 100. [%]	80.
wv	float	positive value [kg/m <sup>2</sup> ]	40.
Visib	float	positive value [km]	10.
WindSp24	float	positive value [m/s]	3.
cski	float	0. .. 2.0	1.
rski	float	0. .. 1.0	0.
cloud	float	0. .. 1.0	0.
comp1	[ATM_COMP]		
comp2	[ATM_COMP]		
comp3	[ATM_COMP]		
comp4	[ATM_COMP]		

Table 1.5: ATM.IN initialisation structure

Atmospheric components (see also paragraph 1.2.1) are initialised through a string array of the form [n,2] containing n lines of assignments in two columns. The first column is the name of the member and the second its value. Note that all member values are inserted as string and that the order of the rows is not important. For instance, COMP\_AER\_OPAC can be initialised through the string array in Table 1.6.

String array ATM_COMP	
First col.	Second col.
'aer_type'	'MARPL70'
'rh'	'80.'
'vls'	'OPAC_01'
'a865'	'0.10'

Table 1.6: ATM\_COMP initialisation string array

**GRID\_IN** structure contains definition of the angular directions for which we want the radiance to be computed. *Azi* is the azimuthal angle referred to North direction and clockwise, while *Zen* is the *cosine* of the zenithal angle, negative for upward radiances. Azimuthal angles must always be provided as float array (in the example in Table 1.7 an equi spaced distribution with  $\phi=30^\circ$  is used), while the zenithal angle can be defined either through a list of values or setting the flag *equi* to 1 (i.e. equi spaced) and entering the number of angles *Nzen*. In the example, a uniform sequence of 14 values within -1. and 1. is used.

Structure GRID_IN			
Member	Type	Values allowed	Example
Azi	*float	0. ... 360	[0.,30.,60.,90.,120.,150.,180.]
Equi	boolean	0 or 1	1
Nzen	int	positive value	14
Zen	*float	-1. .. 1.	[]

Table 1.7: GRID\_IN initialisation structure

**LOCAT** object is initialised like **SUN** through a multiple assignment string delimited by square brackets. Detailed description of each member is in Table (1.8).

String LOCAT			
Member	Type	Values allowed	Example
LAT	float	-90. .. 90.	40.1
LON	float	-180. .. 180.	50.2
JDAY	integer	1 .. 366	121
HH	float	0. .. 24.00	12.34

Table 1.8: LOCAT initialisation string

**BOTTOM** object is initialised as **LOCAT** and **SUN**. See also (1.4.6) for the member description and the example in Table 1.9.

String BOTTOM			
Member	Type	Values allowed	Example
Type	int	0,1,2	2
File	string	valid filename	"GreenAlgae.txt"
Refl	float	0. .. 1.	0.2

Table 1.9: BOTTOM initialisation string

*WAT* object initialisation is performed through the structure **WAT\_IN** as in Table (1.10). Note that in this case the *Name* field is crucial to generate one of the water derived classes described in (1.3.2) and that the **WAT\_COMP** must be assigned accordingly.

Structure WAT_IN			
Member	Type	Values allowed	Example
Name	string	'WAT_CONST' 'WAT_CASE1' 'WAT_CASE2'	'WAT_CASE2'
Isrc	ma-string	see tab. 1.11	
comp1	{WAT_COMP}	see tab. 1.12	
comp2	{WAT_COMP}	see tab. 1.12	
comp3	{WAT_COMP}	see tab. 1.12	
comp4	{WAT_COMP}	see tab. 1.12	

Table 1.10: WAT\_IN initialisation structure

Internal sources are activated again using a multiple assignment string, whose members are detailed in Table (1.11). In case an assignment is missing the default value is used (0). *COMPCHLA* represents the index of the component that provides the Chlorophyll concentration profile and in the current implementation must always be set to 2. The logic of the flags is explained in Mobley and Sundman (2000b, pg. 47).

String ISRC			
Member	Type	Values allowed	Example
BIOLUM	boolean	0,1	0
CHLFLU	boolean	0,1	0
CDOMFLU	boolean	0,1	0
RAMAN	boolean	0,1	0
COMPCHLA	integer	1 .. ncomp	0

Table 1.11: ISRC initialisation string

As for **ATM\_COMP**, a **WAT\_COMP** object is created on the basis of a string array of the form [n,2]. The first column contains the **COMP** member name (e.g. 'phf' for the phase function) and the second the ma-string to be used for creation and initialisation of this member (see Table 1.12).

String Array WAT_COMP	
First col.	Second col.
'id'	string
'amod'	ma-string
'bmod'	ma-string
'phf'	ma-string
'prof'	ma-string

Table 1.12: WAT\_COMP initialisation string array

**OLS\_IN** object is initialised through the structure represented in Table (1.13). It contains the name of the OLS object, type of layer structure (see par. 1.4.5) and actual depths.

Structure OLS_IN			
Member	Type	Values allowed	Example
Name	string	any string	'OLS_01'
Type	integer	0 , 1	0
depths	*float	positive value	

Table 1.13: OLS\_IN initialisation structure

The wavelengths to be used for the computation are simply defined by the structure **WAVE\_IN** in Table (1.14), providing a symbolic name and the list of wavelengths in nanometer.

Structure WAVE_IN			
Member	Type	Values allowed	Example
Name	string	any string	'WAVE_01'
Wave	*float	250. .. 40000	412.

Table 1.14: WAVE\_IN initialisation structure

## 1.6 List of IDL files

- RTM.TOP\_define.pro: defines **SYSTEM** class, its methods and structures for initialisation.
- RTM.GEN\_define.pro: defines general classes and their methods as in 1.1.
- RTM.ATM\_define.pro: defines **ATM** class and its methods as in 1.2.
- RTM.WAT\_define.pro: defines **WAT** class and its methods as in 1.3.
- RTM.OTH\_define.pro: defines the remaining classes and their methods as in 1.4.
- RTM.Display.pro: routines to display RT computation input (IOPs) and output (Radiance, Irradiance, ...).
- RTM.Tool.pro: utility routines to process input/output files, handle IDL structures, ... .



## Chapter 2

# Interface to FEM (Finite Element Method)

FEM is a finite-element method applied to solving the radiative-transfer equation in a layered medium with a change in refractive index, as described by Barbara Bulgarelli, Viatcheslav B. Kisselev and Laura Roberti and described in Bulgarelli et al. (1999).

The 'user version' of the code has been used in the current work, which is a version reading directly from input files the optical properties of each homogeneous layer in atmosphere and water (i.e. above and below the refractive index discontinuity). These properties are the optical depth, single scattering albedo and Legendre associated coefficients of the scattering phase function.

The main issue to interface the water-atmosphere coupled *SYSTEM* defined in Chapter 1 with the FEM code is thus to define a set of layers in atmosphere and water and to compute for every layers the above optical quantities. Section 2.1 describes this process.

The rest of the activity consists in writing input files for the FEM executable, launching the code from IDL and reading/analysing the output produced.

An example of computation on a simple Case 1 water is presented in Section 2.2.

## 2.1 System to FEM interface

**SYSTEM** object does not contain any vertical layer definition: both in atmosphere and water, components are distributed according to continuous vertical profiles from the surface downward in water and upward in atmosphere. A new class **VLS** (Vertical Layer Structure) is defined to set height levels in atmosphere and water, and is displayed in fig. (2.1).

VLS
Type
Nz
Natm
Nwat
Zatm
Zwat

Figure 2.1: Vertical Layer Structure class

*Type* is an integer that sets the levels type as geometrical (0) or optical (1). In the first and default case, atmospheric levels are defined in km from the Top downward to the surface and water levels in m from the surface downward. For a correct processing, it is recommended to set the last level in atmosphere and the first in water as 0. *Nz* is the overall number of levels, i.e. the sum of levels in water (*Nwat*) and in atmosphere (*Natm*), *Zatm* and *Zwat* are two float arrays of levels in the units specified above.

### 2.1.1 Atmosphere Optical Properties

Methods are defined for every atmospheric component to provide optical thickness  $\tau(k)$ , single scattering albedo  $\omega(k)$  and phase function Legendre associated coefficients  $X_{cof}^j(k)$  for each layer  $k$  delimited by a couple of adjacent VLS levels. Obviously the total number of layers  $N_{lyr}$  is  $N_{atm} - 1$ .

- **Ozone** : it is only absorbing, therefore

$$\omega_{O_3}(k) \equiv 0. \quad (2.1)$$

$$X_{cof_{O_3}}(k) \equiv 0. \quad (2.2)$$

The total optical thickness associated to the ozone column ( $\tau_{O_3}$ ) is computed from the total column amount in DBU ( $O_{3DBU}$ ) and tabulated wavelength-dependent conversion coefficient  $K_{O_3}$  :

$$\tau_{O_3}(\lambda) = O_{3DBU} \cdot K_{O_3}(\lambda) \quad (2.3)$$

$K_{O_3}$  are retrieved from '[http://oceancolor.gsfc.nasa.gov/DOCS/RSR/Nicolet\\_o3\\_abs.dat](http://oceancolor.gsfc.nasa.gov/DOCS/RSR/Nicolet_o3_abs.dat)' and derived from Nicolet (1981, Table 13). Optical thickness is 'distributed' over the different layers according to ozone vertical profile  $O_{3ppmv}$  from AGFL meteorological model US76 (see, e.g., Thomas and Stamnes 1999).

$$\tau_{O_3}(k) = \tau_{O_3} \cdot \frac{O_{3ppmv}(k)}{\sum_{i=1}^{N_{lyr}} O_{3ppmv}(i)} \quad (2.4)$$

where  $N_{lyr}$  is the total number of layers and  $O_{3ppmv}(k)$  is computed as arithmetic average of the concentrations at the top and bottom levels of the layer.

- **Aerosols** : aerosol properties are computed differently if the **COMP\_AER** or **COMP\_AER\_OPAC** model is selected.

1. **COMP\_AER**: Ångström law is used to define optical thickness at a generic wavelength:

$$\tau_{aer}(\lambda) = \tau_{865} \cdot \left(\frac{\lambda}{865}\right)^\nu \quad (2.5)$$

The total optical thickness is assigned to the layers according to :

$$\tau_{aer}(k) = \tau_{aer} \cdot (e^{-0.5 \cdot z_i} - e^{-0.5 \cdot z_{i+1}}) \quad (2.6)$$

where  $z_i$  and  $z_{i+1}$  are the bottom and top levels of the layer  $k$ .

Single scattering albedo parameter is assumed to be 1 by default (non absorbing aerosol) but can be modified by assigning a value to **COMP\_AER** *ssa* member.

Phase function Legendre associated coefficients are computed directly by **PHF** dedicated methods.

2. **COMP\_AER\_OPAC**: a formatted dataset of aerosol properties has been produced from the OPAC tool. For the OPAC pre-defined aerosol types (Maritime clean/polluted/tropical, Continental clean/average/polluted, Urban and Desert) at different relative humidity values (50/70/80/90/95/98/99%) and for different wavelengths,  $\tau$ ,  $\omega$  and  $X_{cof}^j$  are pre-computed and stored in binary files. The same has been done for the 'background' aerosol in troposphere and stratosphere. After loading the table corresponding to a given aerosol type, relative humidity and wavelength, the only computation performed here is to 'rescale' the total aerosol optical thickness to the imposed value at 865 nm. This is done modifying the optical thickness in the boundary layer as follows:

$$\tau'_{bnd}(865) = \tau'_{tot}(865) - \tau_{str}^{OPAC}(865) - \tau_{tro}^{OPAC}(865) \quad (2.7)$$

where  $\tau'_{tot}(865)$  is the value imposed to the total optical thickness at 865 nm and read from the **COMP\_AER\_OPAC** *a865* member.  $\tau'_{bnd}(865)$  is the imposed value of optical thickness in the boundary layer used to compute a correction coefficient  $c_{fact}$  as:

$$c_{fact} = \frac{\tau'_{bnd}(865)}{\tau_{bnd}^{OPAC}(865)} \quad (2.8)$$

that is applied to the actual wavelength:

$$\tau'_{bnd}(\lambda) = \tau_{bnd}^{OPAC}(\lambda) \cdot c_{fact} \quad (2.9)$$

- **Rayleigh scattering** : in this case it is by definition

$$\omega_{Ray}(k) \equiv 1. \quad (2.10)$$

Rayleigh total optical thickness  $\tau_{Ray}$  as function of the wavelength is computed by the Hansen-Travis formula (Hansen and Travis 1974), rescaled through the actual Surface Pressure:

$$\tau_{Ray} = 0.008569 \cdot \lambda^{-4} \cdot (1 + 0.0113 \cdot \lambda^{-2} + 0.00013 \cdot \lambda^{-4}) \cdot \frac{P[mbar]}{1013.25} \quad (2.11)$$

Total optical thickness is 'distributed' over the different layers using the air density from AGFL meteorological model US76 profile, by the formula:

$$\tau_{Ray}(k) = \tau_{Ray} \cdot \frac{\tilde{\rho}_{air}(k) \cdot \Delta z_k}{\sum_{i=1}^{nz} \tilde{\rho}_{air}(i) \cdot \Delta z_i} \quad (2.12)$$

where  $\tilde{\rho}_{air}(k)$  is the log-average air density for the layer k and  $\Delta z_k$  is its depth. Rayleigh phase function Legendre coefficients are directly assigned as :

$$X_{cof}^1 \equiv 1. \quad (2.13)$$

$$X_{cof}^3 \equiv 0.5 \quad (2.14)$$

- **Oxygen** : computation is done in the same way as for Ozone, but the oxygen is 'rescaled' using Surface Pressure instead of a total column amount.

$$\tau_{O_2} = K_{O_2}(\lambda) \cdot \frac{P[mbar]}{1013.25} \quad (2.15)$$

$$\omega_{O_2}(k) \equiv 0. \quad (2.16)$$

where  $K_{O_2}$  is the oxygen total optical thickness for standard surface pressure and single scattering albedo is set to 0 as the scattering from oxygen is taken into account by the Rayleigh component.

### • Composite Atmosphere

On the basis of the above computed quantities, overall optical properties are provided by the following equations:

$$\tau_t(k) = \sum_{i=1}^{N_{comp}} \tau_i(k) \quad (2.17)$$

$$\omega_t(k) = \frac{\sum_{i=1}^{N_{comp}} \omega_i(k) \cdot \tau_i(k)}{\sum_{i=1}^{N_{comp}} \tau_i(k)} \quad (2.18)$$

$$X_{cof_t}^j(k) = \frac{\sum_{i=1}^{N_{comp}} X_{cof_i}^j(k) \cdot \omega_i(k) \cdot \tau_i(k)}{\sum_{i=1}^{N_{comp}} \omega_i(k) \cdot \tau_i(k)} \quad (2.19)$$

where t subscript means 'total',  $N_{comp}$  is the number of atmospheric components and j represents the order of the Legendre coefficient.

### 2.1.2 Water Optical Properties

Computation of optical properties for water is more straightforward than for atmosphere, as every water component is defined through an absorption and scattering model, a scattering phase function and a vertical profile. **A\_MOD** and **B\_MOD** include methods to provide directly  $a$  and  $b$  coefficients, once component concentration and wavelength are defined. Thus optical properties are computed as follows:

$$\tau(k) = a(k) + b(k) \quad (2.20)$$

$$\omega(k) = b(k)/(a(k) + b(k)) \quad (2.21)$$

A further computation is needed if the phase function is extremely forward peaked and a truncation has been applied to compute Legendre coefficients. In this case, the first coefficient differs from 1., and its value is used to compute a correction value  $a_{corr}$  as (see Wiscombe 1977):

$$a_{corr} = 1 - X_{coef}^0. \quad (2.22)$$

that is applied to the optical thickness and the single scattering albedo:

$$\tau'(k) = \tau(k) \cdot (1. - \omega(k) \cdot a_{corr}) \quad (2.23)$$

$$\omega'(k) = \omega(k) \cdot \frac{1 - a_{corr}}{1 - \omega(k) \cdot a_{corr}} \quad (2.24)$$

PHF returns directly the Legendre associated coefficients.

Total optical properties for a water layer  $k$  are computed applying the same formula as for the atmosphere (see eq. 2.17 to 2.19).

### 2.1.3 Band averaged quantities (for SeaWiFS)

FEM-IDL allows the use of bandpass averaged quantities instead of wavelength dependent variables, which is suitable for the simulation of instrument bands behaviour with better results than using central wavelength approximation.

This approach has been implemented for SeaWiFS bands by adding a flag 'SeaWIFS' in the Context definition (see 2.3) and modifying the methods involved in the variable retrieval.

The variables that can be defined as band averaged and the methods responsible for their computation are shown in Table 2.1.

Values for SeaWiFS are taken from [http://oceancolor.gsfc.nasa.gov/RSR\\_tables.html](http://oceancolor.gsfc.nasa.gov/RSR_tables.html).

Quantity	Method	Name in SeaWiFS web page
Pure Water Abs. Coeff	A.PeF::FEMGet.a	$a_w$
Pure Water Sca. Coeff	B.SeB::FEMGet.b	$bb_w$ (*)
Ozone Absorption	COMP_O3::FEMGetVar	$k_{oz}$
Rayleigh Opt. Thick.	COMP_MOL::FEMGetVar	$\tau_{aR}$
Solar Irradiance	(**)	$*F$

Table 2.1: Band averaged quantities

(\*) Pure Water scattering coefficient is computed as  $b_{pw} = bb_w \cdot 2$

(\*\*) Solar Irradiance band averaged values are used directly setting the SUN *Edtot* member in system definition (see 1.4.1).

## 2.2 Example

This section shows a simple water Case 1 example, partially similar to one reported for Hydrolight in 3.2. *'top\_ref.bat'* is an IDL batch file that contains all actions necessary to run the example.

```
@def_env.bat

; -----
; ****    1. Create system object
; -----

@sys_ref.bat

; -----
; ****    2. Run FEM
; -----

RID='Ref'

sys -> FEM_input, Ctx, vls, str_in1, str_in2, str_mom
st  = FEM_PP_RUN(Ctx,str_in1,str_in2,str_mom,RID=rid)

; -----
; ****    3. Load results
; -----

st = FEM_AC_READ_OUT3(Ctx, str_out,RID=rid)

; -----
; ****    4. Plot IOPs
; -----

sys -> FEM_Input, Ctx, vls, $
                        str_in1, $
                        str_in2, $
                        str_mom, $
                        DBG_ATM=1, $
                        DBG_WAT=1, $
                        EPS=1

; -----
; ****    5. Plot computed Radiances
; -----

FEM_DD_RADIANCE, str_out, $
                  DEPTH=0, $           ; Top of Atmosphere
                  RID=rid, $
                  THETA=[-60.],$       ; Add dotted line
```

```

EPS=1

FEM_DD_RADIANCE, str_out, $
DEPTH=1, $ ; Above Surface
RID=rid, $
THETA=[-60.,-120.],$ ; Add dotted lines
EPS=1

FEM_DD_RADIANCE, str_out, $
DEPTH=2, $ ; Below Surface
RID=rid, $
THETA=[-139.74,139.74],$ ; Add dotted lines
EPS=1

FEM_DD_RADIANCE, str_out, $
DEPTH=6, $ ; At 10 m depth
RID=rid, $
THETA=[-139.74],$ ; Add dotted lines
EPS=1

FEM_DD_RADIANCE, str_out, $
DEPTH=10, $ ; At 20 m depth
RID=rid, $
THETA=[-139.74],$ ; Add dotted lines
EPS=1

FEM_DD_RADIANCE, str_out, $
DEPTH=17, $ ; At 40 m depth
RID=rid, $
THETA=[-139.74],$ ; Add dotted lines
EPS=1

```

'*def\_env.bat*' sets up the IDL environment by compiling all the needed modules, for both SYS-IDL and FEM-IDL. This script is available at 2.3.1.

'*sys\_ref.bat*' defines the appropriate physical system and is discussed in 2.2.1.

FEM code is driven by the IDL FEM\_PP\_RUN routine; details about selection and use of FEM executable can be found in 2.2.2.

At the end of the run, results are loaded and displayed by the routine FEM\_DD\_RADIANCE as explained in 2.2.3.



## 2.2.1 Input description

The sun is located at a solar zenith angle of  $60^\circ$  and provides a spectral irradiance at the top of atmosphere of  $1 \text{ W m}^{-2} \text{ nm}^{-1}$  on a surface perpendicular to the Sun's rays. The atmosphere contains absorbing gases ( $O_2$  and  $O_3$ ), distributed according to a standard profile U76, and aerosols.  $O_3$  concentration is rescaled to a total column amount of 350 DBU while Surface Pressure is not defined for  $O_2$  component and the default value is used. Aerosol particles have a total optical thickness of 0.05 at 865 nm and follow Ångström law (with  $nu = 1.0$ ). Aerosol scattering phase function is assumed to be a TTHG with  $g_1 = 0.85$ ,  $g_2 = 0.7$  and  $\alpha = 0.95$ . Rayleigh scattering is also taken into account by a fourth component.

Pure water properties are represented by Pope and Fry model for absorption and by Smith and Baker model for scattering (refer to sect. 1.3 for details and references). Chlorophyll particles are distributed according to a gaussian profile as in 1.1.1 with parameters  $\text{sig}=9. \text{ m}$ ,  $A=6.3831 \text{ mg m}^{-3}$ ,  $C=.2 \text{ mg m}^{-3}$ ,  $xm=17. \text{ m}$ . A\_BRI and B\_MeM models are used for the absorption and scattering coefficients, while the Petzold phase function is assumed for particles.

Computation is performed at  $\lambda = 510 \text{ nm}$ , using 14 layers in atmosphere and 22 in water. Output azimuth angles are defined to be regularly distributed between 0 and 180 degree, at 45 degree step, while zenith angles are set to 18 values in the range 0 to 180 degree with uniform step.

Here below the script '*sys\_ref.bat*', defining the SYS\_IN structure used for *SYSTEM* initialisation, is reported.

```
; -----
; ****    Assign sys_in values
; -----
;

RTM_TOP_DEFINE_CLASSES, sys_in

; -----
; ****    GEN values
; -----

sys_in.id      = 'Reference_case'
sys_in.desc    = 'Reference case'

sys_in.sun     = '[SUN: ZEN=60.,AZI=0., EDTOT=1.]'
sys_in.bottom  = '[BOTTOM: ]'

sys_in.wave.name    = 'WAVE'
sys_in.wave.wave    = '[510.]'

; -----
; ****    GRID
; -----

azi              = PTR_NEW(FLTARR(5))
(*azi)           = [0.,45.,90.,135.,180.]
sys_in.grid.azi   = azi
```

```

sys_in.grid.nzen      = 18
sys_in.grid.equi      = 1
zen                   = PTR_NEW(FLTARR(sys_in.grid.nzen))

(*zen)(0)             = [-1.000000, -0.982973, -0.932472, -0.850217, -0.739009, $
                        -0.602635, -0.445738, -0.273663, -0.092268,  0.092268, $
                        0.273663,  0.445738,  0.602635,  0.739009,  0.850217, $
                        0.932472,  0.982973, 1.0000]

sys_in.grid.zen       = zen

; -----
; ****      WAT
; -----

sys_in.wat.name       = 'WAT_CASE1'

sys_in.wat.comp1(0,1) = ''
sys_in.wat.comp1(1,1) = '[A_PeF: ]'
sys_in.wat.comp1(2,1) = '[B_SeB: ]'
sys_in.wat.comp1(3,1) = '[PHF_FILE: NAME="PHF_pw", TYPE="Legendre"]'

sys_in.wat.comp2(0,1) = ''
sys_in.wat.comp2(1,1) = '[A_BRI: ]'
sys_in.wat.comp2(2,1) = '[B_MeM: c550=0.416, e550=0.766 ]'
sys_in.wat.comp2(3,1) = '[PHF_FILE: NAME="PHF_petzold", TYPE="Legendre"]'
sys_in.wat.comp2(4,1) = '[GAUSSIAN: sig=9. , A=6.3831, C=.2, xm=17.]'

; -----
; ****      ATM
; -----

; COMP_03

prof = '[SERIE: file="/home/clerima/FEM/data/Ozone_U76.dat"]'
sys_in.atm.comp1(0,0) = 'prof' & sys_in.atm.comp1(0,1) = prof
sys_in.atm.comp1(1,0) = '03'  & sys_in.atm.comp1(1,1) = '350.0'

; COMP_AER

sys_in.atm.comp2(0,0) = 'a'    & sys_in.atm.comp2(0,1) = '0.05'
sys_in.atm.comp2(1,0) = 'nu'  & sys_in.atm.comp2(1,1) = '1.0'
sys_in.atm.comp2(2,0) = 'PHF'
sys_in.atm.comp2(2,1) = "[PHF_TTHG: as=0.95, g1=0.85, g2=0.7]"

; COMP_MOL

sys_in.atm.comp3(0,0) = 'PHF' & sys_in.atm.comp3(0,1) = "[PHF_RAYL: ]"

```

```

sys_in.atm.comp3(1,0) = 'p0' & sys_in.atm.comp3(1,1) = '1013.0'

; COMP_02

prof = '[SERIE: file="/home/clerima/FEM/data/Oxigen.dat"]
sys_in.atm.comp4(0,0) = 'prof' & sys_in.atm.comp4(0,1) = prof

; -----
; **** Initialise system
; -----

sys = OBJ_NEW("SYSTEM")
sys -> Initial, sys_in

; -----
; **** VLS
; -----
vls = {VLS}
vls.type = 0 ; geometrical

vls.nz = 38

vls.natm = 15 ; number of levels
zatm = PTR_NEW(FLTARR(15))
(*zatm)(*) = [60.,50.,40.,35.,30.,25.,20.,15.,$
10.,6.,3.,2.,1.,0.5, 0.]
vls.zatm = zatm

vls.nwat = 23 ; number of levels
zwat = PTR_NEW(FLTARR(23))

(*zwat)(*) = [ 0.0, 1.0, 2.0, 3.5, 5.0, $
7.5, 10.0, 12.5, 15.0, 17.5, $
20.0, 22.5, 25.0, 27.5, 30.0, $
32.5, 35.0, 40.0, 50.0, 60.0, $
70.0, 80.0, 90.0]

vls.zwat = zwat

```

## 2.2.2 Run of the example case

In order to run FEM, 3 ASCII input files are required, whose organisation is described in '*femwat.txt*' file, distributed together with the FEM User Version. FEM-IDL defines a structure corresponding to each file (STR\_IN1, STR\_IN2 and STR\_MOM), together with routines to write the files in the appropriate format.

Once a *SYSTEM* object is created, as in 2.2.1, its FEM input method is used to transfer values to structures STR\_IN1, STR\_IN2 and STR\_MOM, which are then passed to FEM\_PP\_RUN, which does the following:

- Writes FEM ASCII input files.
- Calls the FEM executable.

## 2.2.3 Output analysis

Optical properties of every atmosphere/water component cannot be loaded from the FEM output files, as the code receives for each layer only total  $\tau$ ,  $\omega$  and Legendre coefficients, computed from equations 2.17 to 2.19. Therefore the same method used to compute IOPs (system::FEM\_Input) contains keywords to display them, both for water and atmosphere. These keywords (DBG\_WAT and DBG\_ATM) are activated in '*top-ref.bat*' batch file to generate the figures 2.2 to 2.6.

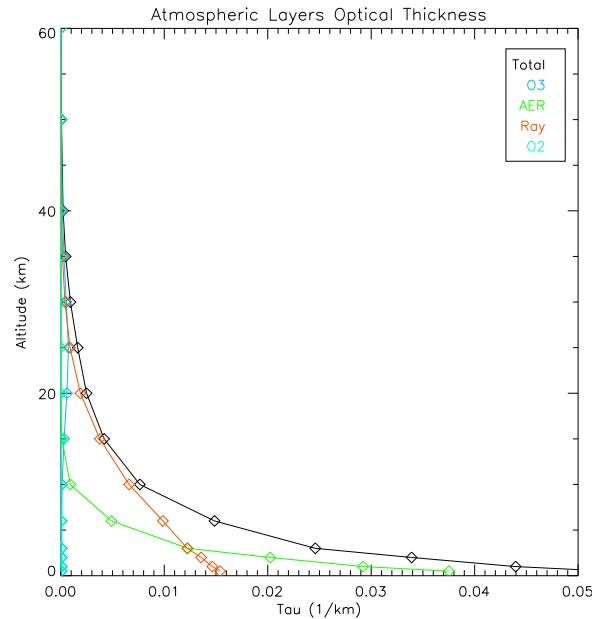


Figure 2.2: Optical Thickness for atmospheric layers

Figure 2.2 represents different contributions to the optical thickness in atmosphere vs. altitude in km. The quantity represented is the optical depth  $\tau_i(k)$ , as computed in 2.1.1, divided by the layer height in km. This is done in order to avoid discontinuities due to the non uniform levels spacing (layers close to surface are thinner than at TOA). Ozone absorption can be discerned, with its maximum around 23 km, while in lower layers Rayleigh scattering and interaction with aerosol dominate.

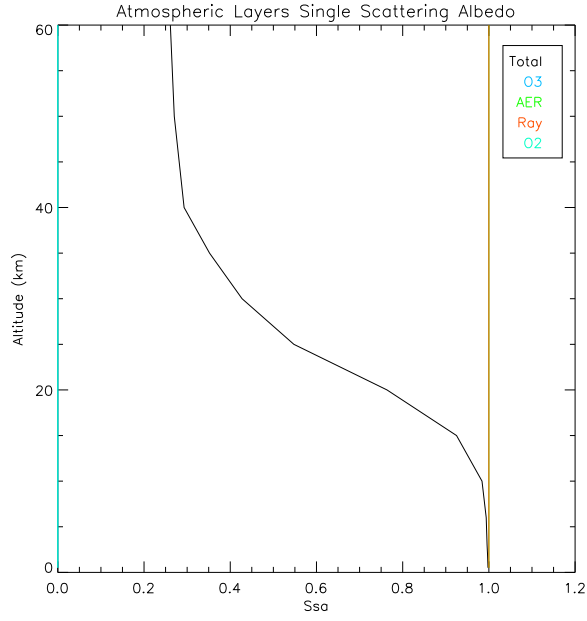


Figure 2.3: Single Scattering Albedo for atmospheric layers

As it can be seen from fig. 2.3, in the present case components are only absorbing ( $O_2$  and  $O_3$ ) or only scattering (aerosols and Rayleigh), and the shape of total Single Scattering Albedo is determined by the different optical thickness of components along the vertical profile (see eq. 2.18).

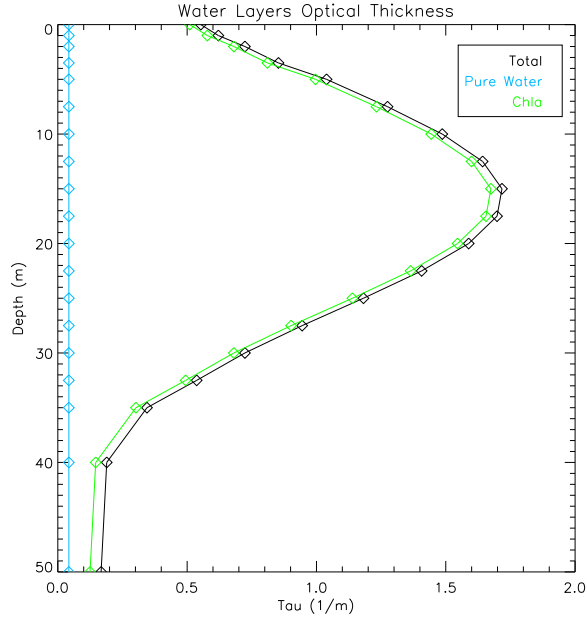


Figure 2.4: Optical Thickness for water layers

In the water body the Chlorophyll gaussian distribution with its maximum at 17 m can be clearly seen in fig. 2.4.

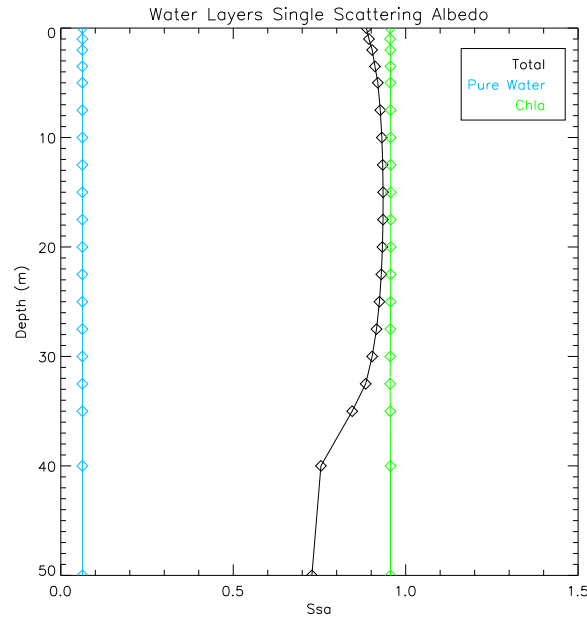


Figure 2.5: Single Scattering Albedo for water layers

Chlorophyll component single scattering albedo varies slightly in water together with the variation of the particle concentration along the profile, as expected. Nevertheless ssa value is always close to 0.95 and the variation cannot be appreciated from fig. 2.5.

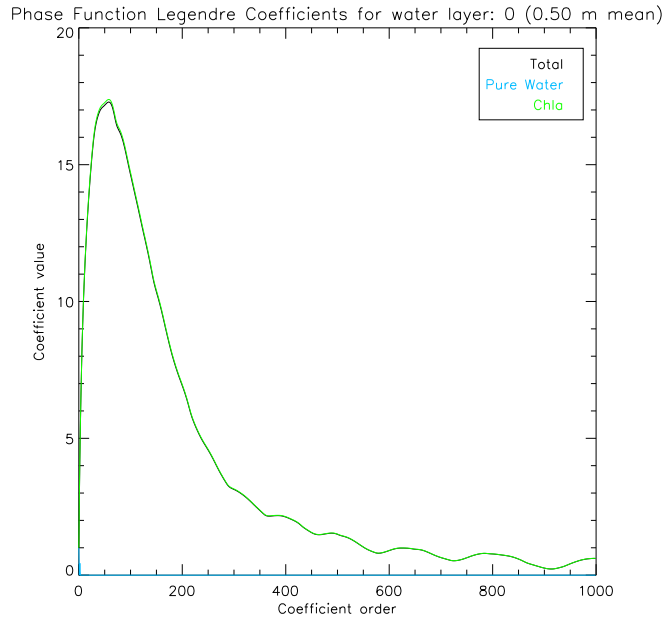


Figure 2.6: Phase Function Legendre Coefficients for first water layer

Phase function Legendre coefficients, computed for the first water layer below the surface, are displayed in fig. 2.6.

In order to understand radiance distribution computed from the FEM code, presented and discussed hereafter, some conventions must be explained.

- Radiance : the FEM code produces as output **only diffuse radiance**, i.e. radiance coming from at least one interaction with particles in water or atmosphere, and not the direct radiance. Therefore, radiance coming directly from the Sun will not appear on plots.
- Viewing direction: it is the direction in which points an hypothetical instrument (or the eye of the observer) and **not** the direction of propagation of the photons.
- Azimuth viewing angle: by definition  $\phi = 0$  refers to the semi-plane of the direct light propagation while  $\phi = 180$  refers to the opposite semi-plane (containing the Sun).
- Zenith viewing angle:  $\theta$  is 0 when looking straight downward at the upcoming radiation.

The convention defined for the viewing angles (see fig. 2.7) is the same as in Mobley (1994, pp. 505-506), but the two opposite azimuthal semi planes ( $\phi = 0$  and  $\phi = 180$ ) are combined differently, making adjacent the value  $\theta = 0$  rather than  $\theta = 180$  (see e.g. fig. 2.8). This is done because the current approach is focused on the remote sensing problem, rather than the in-water light propagation, and in most of the cases only upward radiance at TOA will be analysed with  $\theta$  in a range  $-\theta_{max} \dots +\theta_{max}$ .

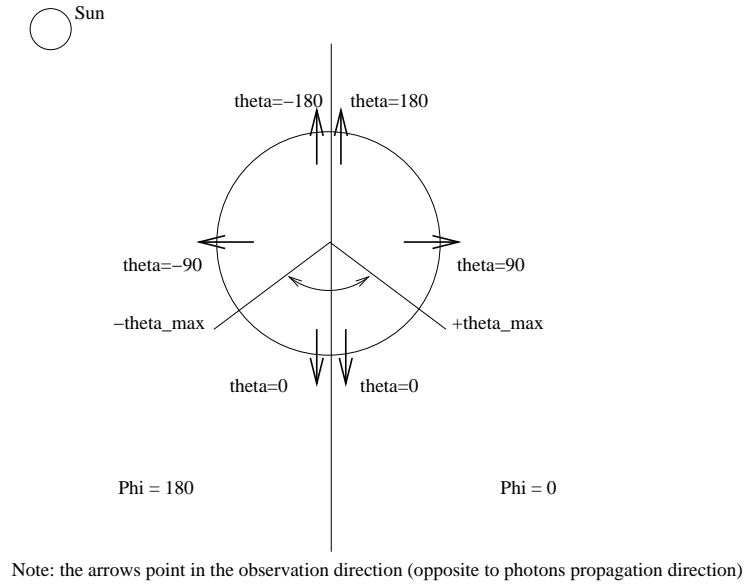


Figure 2.7: Viewing angle description

Radiance distributions in the principal plane, plotted by the routine FEM\_DD\_RADIANCE called several times for different levels in the script file '*top\_ref.bat*', are discussed in the following.

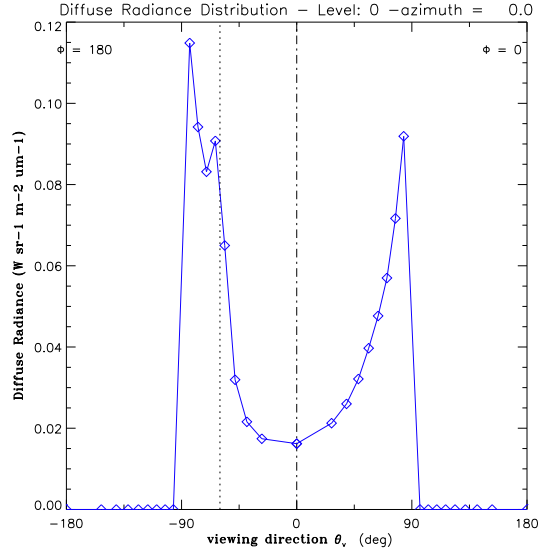


Figure 2.8: Diffuse radiance at TOA

At Top of Atmosphere diffuse radiance is obviously non-zero only in the range  $\theta = -90^\circ$  to  $\theta = 90^\circ$  and has its maximum at these boundary values, as effect of both aerosol and Rayleigh scattering in atmosphere. For  $\theta = -60^\circ$  a peak due to the forward scattered and water reflected radiation can be seen.

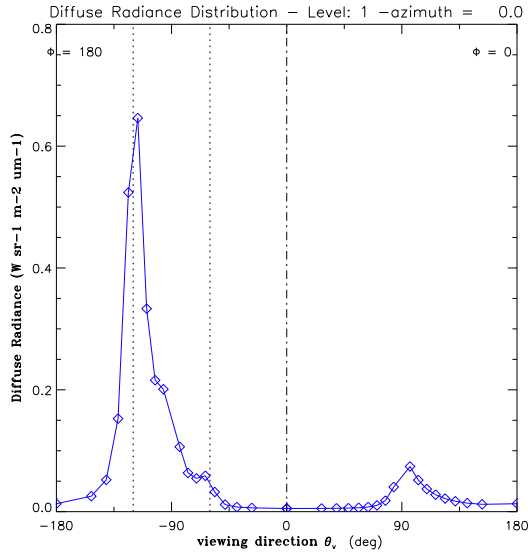


Figure 2.9: Diffuse radiance above water Surface

Figure 2.9 illustrates radiance distribution just above water surface; in the left plane ( $\phi = 180^\circ$ ) two peaks due to forward-scattered radiation can be seen at  $\theta = -120^\circ$  and  $\theta = -60^\circ$ . The first refers to down going radiation, the second to up going radiation after water surface reflection. Diffuse radiation maxima at  $\theta = -90^\circ$  and  $\theta = 90^\circ$  seen in fig. 2.8 are still present.



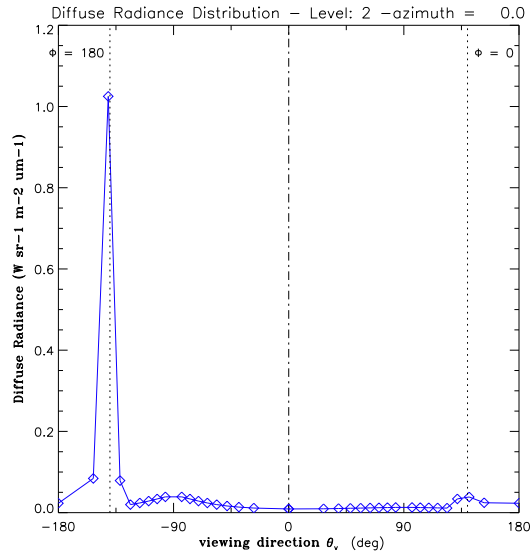


Figure 2.10: Diffuse radiance below water Surface

In figure 2.10 the forward scattered peak is visible near  $\theta = -139.74^\circ$ , which is its propagation direction after refraction. A small peak is also present at the opposite direction, as an effect of the chlorophyll particles back-scattering.

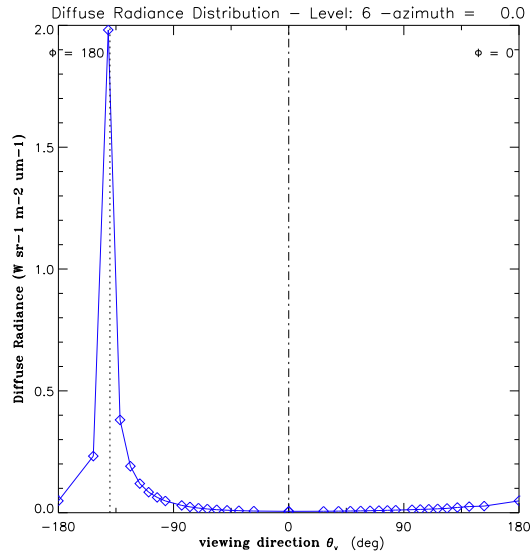


Figure 2.11: Diffuse radiance at 10 m depth

Light propagation in water body is illustrated in figures 2.11 to 2.13. At 10 meter depth, forward scattered radiation around  $\theta = -139.74^\circ$  is stronger than just below surface, due to particles scattering.

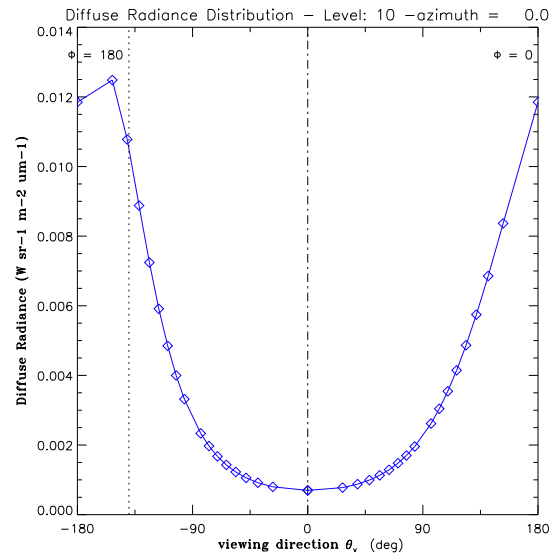


Figure 2.12: Diffuse radiance at 20 m depth

At 20 meter depth (see figure 2.12) the peak due to forward scattering is still visible but much lighter than before, while at 40 meter the radiance distribution is almost symmetrical and radiation is coming only downward.

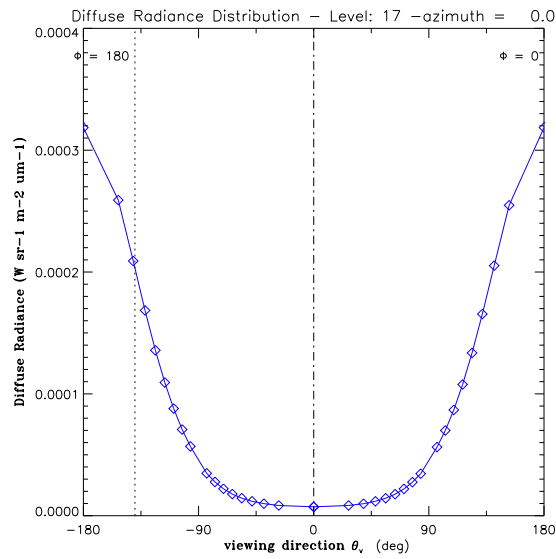


Figure 2.13: Diffuse radiance at 40 m depth

## 2.3 List of IDL files

- FEM\_Methods.pro: methods of the RTM classes (defined in Chapter 1) for FEM adaptation and FEM dedicated classes definition.
- FEM\_Access.pro: routines to read/write input/output files.
- FEM\_Process.pro: routines to drive the FEM code.
- FEM\_Display.pro: routines to display FEM input and outputs.
- FEM\_Tool.pro: define the general environment (Context), which contains directory definition, FEM executable name, error severity levels, satellite specific variables. Initialisation of the Context is done through the files FEM\_Config.icl and def\_SeaWiFS.bat.
- FEM\_Config.icl: definitions for Context initialisation.
- def\_SeaWiFS.bat: SeaWiFS bandpass averaged quantities.

### 2.3.1 IDL environment setting

The attached IDL batch file (*'def\_env.bat'*) performs the following actions:

- Adds SYS-IDL library directory to IDL path.
- Compiles all SYS-IDL files and calls routines to define structures and classes.
- Compiles all FEM-IDL files and calls routines to define structures and classes.
- Creates the general context, containing directories and input/output files definition.

```
!QUIET=1
retall

; Get RTM IDL library dir and add to path
rtmidl=GETENV('RTMIDL')
pos = STRPOS(!path,rtmidl)
IF (pos LT 0) THEN BEGIN !path = !path+':'+rtmidl & PRINT, rtmidl+' added to path'

; Get FEM IDL library dir and add to path
femidl=GETENV('FEMIDL')
pos = STRPOS(!path,femidl)
IF (pos LT 0) THEN BEGIN !path = !path+':'+femidl & PRINT, femidl+' added to path'

; Compile IDL global library

.run ~/IDL/global_rout.pro

; Compile RTM sources and define structures/classes
.run RTM_Tool.pro
.run RTM_Display.pro
```

```

.run RTM_GEN_define.pro
.run RTM_OTH_define.pro
.run RTM_WAT_define.pro
.run RTM_ATM_define.pro
.run RTM_TOP_define.pro

RTM_GEN_DEFINE_CLASSES
RTM_WAT_DEFINE_CLASSES
RTM_OTH_DEFINE_CLASSES
RTM_ATM_DEFINE_CLASSES
RTM_TOP_DEFINE_CLASSES

;   Compile FEM sources and define structures/classes
.run FEM_Tool.pro
.run FEM_Access.pro
.run FEM_Process.pro
.run FEM_Display.pro
.run FEM_Methods.pro

FEM_DEFINE_CLASSES
st  = FEM_AC_DEFINE_STRUCT()

;   Create the context
st  = FEM_TL_GET_CONTEXT(Ctx,DIR='~/FEM/IDL/')

```

# Chapter 3

## Interface to Hydrolight 4.1

Hydrolight 4.1 is a radiative transfer numerical model that computes radiance distribution and derived quantities for natural water bodies from Curtis D. Mobley (Mobley 1994). This model solves the time-independent radiative transfer equation to obtain the radiance distribution within and leaving any plane-parallel water body. The source code is written entirely in FORTRAN; input and output files are ASCII files. The current IDL implementation drives the executable code by writing the input files without the use of the "front-end" program available as an User Interface on Microsoft Windows systems and as a text-based "question-and-answer" program for Unix/Linux platforms, till Hydrolight 4.0 release.

The main issue to interface the water-atmosphere coupled system defined in Chapter 1 with the Hydrolight 4.1 code is to transfer the information stored in *SYSTEM* into the Hydrolight input file. An example of this input file is available in 3.2.2. As it can be seen it contains only values to be read from the executable code without any comment line or keyword in assignment. File detailed description can be found in Mobley and Sundman (2000b, App.A).

The approach followed by Hydrolight is to select during the initialisation phase some routines to be used in run-time and to compile the executable code for every run. On one hand this process makes the code extremely flexible and is suitable for a single run on many wavelengths. On the other hand it is not convenient when the same routines are used for a lot of runs that differ only for the input parameters. In order to avoid this compilation, three water cases have been identified ('WAT\_CONST', 'WAT\_CASE1', 'WAT\_CASE2') and three corresponding executables compiled once for all and called respectively maincode\_C.exe, maincode\_1.exe and maincode\_2.exe. The IDL driver selects one of them in run time according to the *SYSTEM* defined.

The example 2 described in Mobley and Sundman (2000b, pg. 54) is presented in Section 3.2.

### 3.1 System to Hydrolight interface

Hydrolight 4.1 program is focused on the water part of the system, and the atmosphere is considered only to compute the irradiance reaching the water surface but the radiative equation is not solved for the atmospheric layers. This is done in Hydrolight by using different models of 'sky' (see Mobley and Sundman 2000b, pg. 43-44). Thus a new IDL class, displayed in fig. (3.1) is inserted to represent these sky models. This class is defined here, as part of the Hydrolight-IDL routines, rather than at '*System level*', as it is specific to the Hydrolight code.

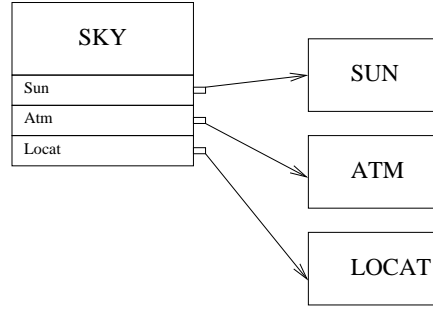


Figure 3.1: Sky class

As it can be seen from fig. (3.1) an object of **SKY** class contains *SUN*, *LOCAT* and *ATM* objects, and it can therefore access the whole information stored in them. **SKY** methods have been developed to 'translate' the atmospheric and solar properties (accounting also for the location selected) into the format Hydrolight expects.

**SKY** derived classes corresponding to the three models implemented in Hydrolight are represented in fig. (3.2).

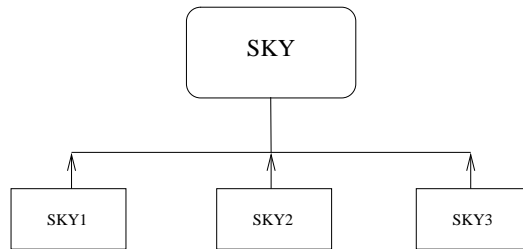


Figure 3.2: Sky derived class

A full description of the three models can be found in Mobley and Sundman (2000a, 2.5).

Unlike for FEM (see Chapter 2), no computation of optical properties has to be done by IDL before calling the Hydrolight executable. In order to transfer information from *SYSTEM* to the Hydrolight input file in the proper format, specific methods are written for all relevant classes (like A.MOD, B.MOD, water components, PHF and so on). A list of these methods can be found in 4.3.

## 3.2 Example

This section shows how to run the example 2 from Mobley and Sundman (2000b, 6.2), which is a multi-spectral Case 1 water simulation. All the needed actions are performed through a simple IDL batch file called "top\_UG2.bat" reported hereafter.

@def\_env.bat

```
; -----
; ****      Create system object
; -----
```

@sys\_UG2.bat

```
; -----
; ****      Run Hydrolight 4.1
; -----
```

```
st = HYD_PP_RUN(Ctx, Str_in1,SKYTYPE=2,SYSTEM=sys)
```

```
; -----
; ****      Load results
; -----
```

```
Ctx.RID = 'Ex_UG2'
```

```
st = HYD_AC_READ_DIGITAL (Ctx, str_out)
```

```
; -----
; ****      Plot Water IOPs
; -----
```

```
HYD_DD_WAT_COEFF, str_out, TYPE='a',$      ; Absorption coefficient
                                SURF=1, $    ; Surface (3D) plot
                                COMP=0, $    ; Only total values
                                iWAVE=1, $   ; First wavelength
                                EPS=1        ; Print to EPS file
```

```
; -----
; ****      Plot computed Radiances
; -----
```

```
HYD_DD_RADIANCE, str_out,  TYPE=1, $      ; Diffuse Radiance
                                DEPTH=5, $  ; Output layer
                                LOG=1,  $   ; Log y-axis
                                SURF=1,  $   ; Surface (3D) plot
                                REFL=0,  $   ; Reflectance
                                EPS=0       ; Print to EPS file
```

*'def\_env.bat'* sets up the IDL environment by compiling all the needed modules, for both SYS-IDL and Hydrolight-IDL. This script is available at 3.3.1.

*'sys\_UG2.bat'* defines the appropriate physical system and is discussed in 3.2.1.

Hydrolight 4.1 code is driven by the IDL HYD\_PP\_RUN routine; details about selection and use of Hydrolight executable can be found in 3.2.2.

At the end of the run, results are loaded and displayed in the last part of the script as explained in 3.2.3.



### 3.2.1 Input description

*SYSTEM* object initialisation follows rules and syntax described in Chapter 1, whose knowledge is a pre-requisite to understand actions described in the current section. The attached file (*'sys\_UG2.bat'*) contains comments that should make understandable every definition of *SYS\_IN* members. Therefore only few key points are highlighted here:

- The order of the assignments follows the one of the Windows GUI, as in Mobley and Sundman (2000b, 6.2). The name of the corresponding forms is also reproduced in the comments as much as possible.
- Water components IOP definition is exactly the same as in Hydrolight, even if it seems a lot more complicated here than through the GUI. That is because we decided not to initialise with the default values absorption and scattering models.
- The definition of the atmosphere here is limited to only few scalar quantities (like cloud coverage, wind speed, ...) which are used by the script to define the Air-Water Surface Boundary Conditions.

```
;
; Batch file to define System as in Hydrolight 4.1 User Guide
; Example 2
;

; -----
; ****    Create SYS_IN structure
; -----

RTM_TOP_DEFINE_CLASSES, sys_in

; -----
; ****    Define model to use
; -----

sys_in.model          = 'HYD'

; -----
; ****    Run Identification
; -----

sys_in.id             = 'Ex_UG2'
sys_in.desc           = 'Example 2: A Simulation of Case 1 water'

; -----
; ****    WATER definition
; -----

sys_in.wat.name       = 'WAT_CASE1'

; ** IOP specification for Component 1: pure water
```

```

sys_in.wat.comp1(0,1) = ''
sys_in.wat.comp1(1,1) = '[A_PeF: FILE="pfh2oab.txt"]'
sys_in.wat.comp1(2,1) = '[B_SeB: ]'
sys_in.wat.comp1(3,1) = '[PHF_FILE: NAME="pureh2o.dpf", TYPE="dpf"]'

; ** IOP specification for Component 2: chlorophyll

sys_in.wat.comp2(0,1) = ''
sys_in.wat.comp2(1,1) = '[A_PSM: FILE="../../data/defaults/apstarchl.txt", wref=440]'
sys_in.wat.comp2(2,1) = '[B_POW: wref=550, bref=0.3, m=1.0, n=.62 ]'
sys_in.wat.comp2(3,1) = '[PHF_FILE: NAME="avgpart.dpf", TYPE="dpf"]'
sys_in.wat.comp2(4,1) = '[VPROFILE: file="../../data/examples/chlzdata.txt"]'

; ** Internal Source and Inelastic Scatter Selection

sys_in.wat.isrc      = '[ISRC: BIOLUM=0, CHLFLU=1, CDOMFLU=0, RAMAN=1, COMPCHLA=2]'

; ** Wavelength selection

sys_in.wave.name      = 'WAVE'
sys_in.wave.wave      = '[350., 360., 370., 380., 390., 400., 410., 420., 430., 440.,'+$
                        '450., 460., 470., 480., 490., 500., 510., 520., 530., 540.,'+$
                        '550., 560., 570., 580., 590., 600., 610., 620., 630., 640.,'+$
                        '650., 660., 670., 680., 690., 700.]'

; ** 'Atmosphere' definition (Sky + water surface)

sys_in.atm.csky       = 1.25
sys_in.atm.rsky       = 0.333
sys_in.atm.cloud      = 0.3
sys_in.atm.winddir    = 0.0
sys_in.atm.windspd    = 2.0

sys_in.sun            = '[SUN: ZEN=20.,AZI=0.,EDTOT=1.]'
sys_in.locat          = '[LOCAT :]'

; ** Bottom Boundary Condition

sys_in.bottom         = '[BOTTOM: TYPE=1      ,REFL=.2,FILE="dummy.txt"]'

; ** Output Depths

sys_in.ols.name        = 'OLS'
sys_in.ols.type        = 0
sys_in.ols.depths     = '[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]'

```

```
; -----
; ****   Initialise system
; -----
```

```
sys = OBJ_NEW("SYSTEM")
sys -> Initial, sys_in
```

### 3.2.2 Run of the example case

Once a **SYSTEM** object has been created and initialised as in 3.2.1, it is passed as keyword to the routine HYD\_PP\_RUN, which does the following:

- Writes Hydrolight run-time input file. The file produced for the current example is attached below.
- Calls the Hydrolight executable related to water Case 1 (maincode\_1.exe).

Example 2: A Simulation of Case 1 water

Ex\_UG2

```
0      1      2      1
2      2
1    999.0    0.9990    0.9990
3    440.0    0.9990    0.9990
```

pfh2oab.txt

../data/defaults/apstarch1.txt

```
0      1.0    0.999000    0.999000    0.999000    0.999000
1    550.0    0.300000    1.000000    0.620000    0.000000
```

bstardummy.txt

bstardummy.txt

```
0      0    0.9990    0.9990
1      0    0.9990    0.9990
```

pureh2o.dpf

avgpart.dpf

```
35
350.00  360.00  370.00  380.00  390.00  400.00  410.00  420.00
430.00  440.00  450.00  460.00  470.00  480.00  490.00  500.00
510.00  520.00  530.00  540.00  550.00  560.00  570.00  580.00
590.00  600.00  610.00  620.00  630.00  640.00  650.00  660.00
670.00  680.00  690.00  700.00
```

```
0  1  0  1  2
```

```
2  3    20.00    0.00    0.30
```

```
2.00
```

```
1    0.2000
```

```
0  11    0.00    2.00    4.00    6.00    8.00    10.00    12.00    14.00    16.00
    18.00    20.00
```

pfh2oab.txt

```
1
```

ac9FileDummy.txt

```

ac9FiltDummy.txt
HydroscatDummy.txt
../data/examples/chlzdata.txt
CDOMDummy.txt
dummy.txt
PWProfileDummy.txt
../data/examples/chlzdata.txt

```

### 3.2.3 Output analysis

This section aims at illustrating just some features of the Hydrolight-IDL visualisation routines: for an overall description of IDL functions and related keywords refer to 4.3. Run results are loaded from 'digital' output file by the routine `HYD_AC_READ_DIGITAL()` and stored in the structure `str_out`. This structure is passed to some visualisation routines to display both water IOPs and output radiance/irradiance. Fig. 3.3 displays water absorption coefficient [ $m^{-1}$ ] as a function of the geometrical depth and wavelength.

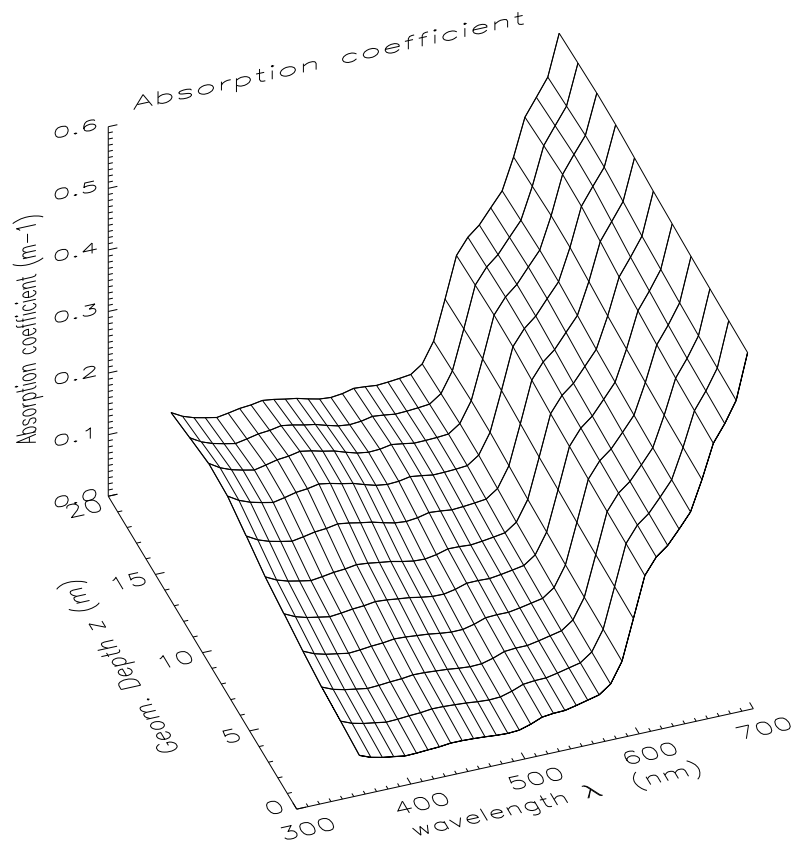


Figure 3.3: Water Absorbing coefficient for Example 2 User Guide

The radiance distribution at a depth of 10 m as a function of the viewing angle and wavelength is represented in Fig. 3.4. Note that the viewing angle convention differs from the one used in Hydrolight User Guide. Here  $\theta = 0$  refers to downward going radiance.

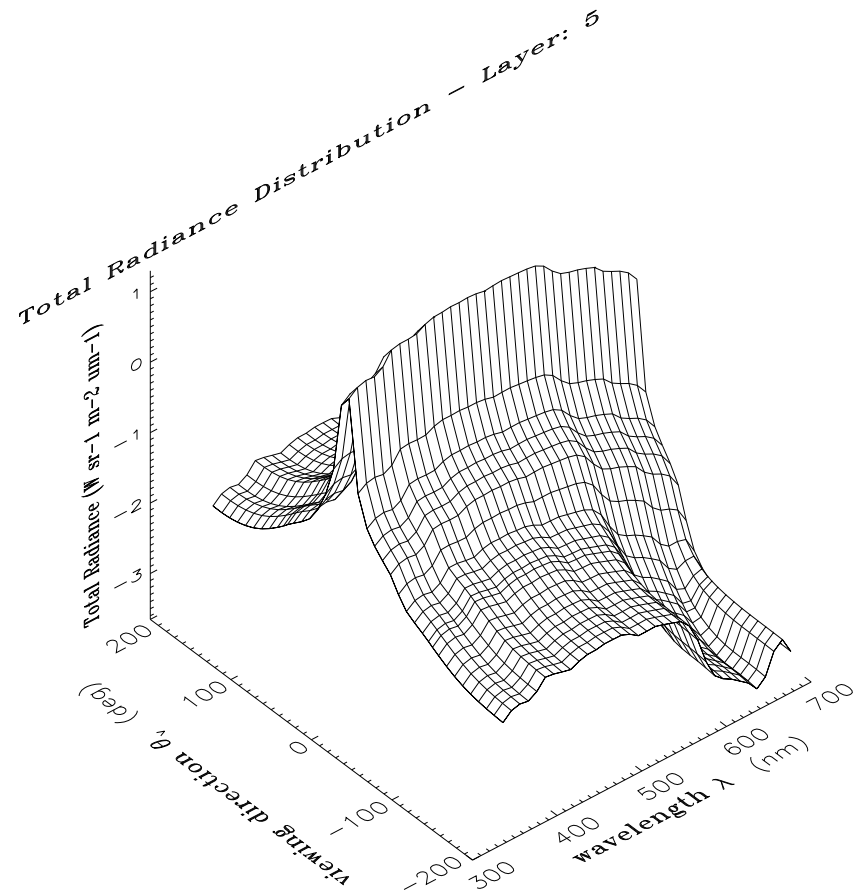


Figure 3.4: Radiance field at 10 m for Example 2 User Guide

### 3.3 List of IDL files

- HYD\_Methods.pro: methods of the SYS classes (defined in Chapter 1) for Hydrolight adaptation and Hydrolight dedicated classes.
- HYD\_Access.pro: routines to read/write input/output files.
- HYD\_Process.pro: routines to drive Hydrolight code.
- HYD\_Display.pro: routines to display Hydrolight input and output.
- HYD\_Tool.pro: define the general environment (Context).

#### 3.3.1 IDL environment setting

The attached IDL batch file (*'def.env.bat'*) performs the following actions:

- Adds SYS-IDL library directory to IDL path.
- Compiles all SYS-IDL files and calls routines to define structures and classes.
- Compiles all Hydrolight-IDL files and calls routines to define structures and classes.
- Creates the general context, containing directories and input/output files definition.

```
retall
!QUIET=1

; Get RTM IDL library dir and add to path
rtmidl=GETENV('RTMIDL')
pos = STRPOS(!path,rtmidl)
IF (pos LT 0) THEN BEGIN !path = !path+':'+rtmidl & PRINT, rtmidl+' added to path'

.run ~/IDL/global_rout.pro

; Compile RTM sources and define structures/classes

.run RTM_Tool.pro
.run RTM_Display.pro
.run RTM_GEN_define.pro
.run RTM_OTH_define.pro
.run RTM_WAT_define.pro
.run RTM_ATM_define.pro
.run RTM_TOP_define.pro

RTM_GEN_DEFINE_CLASSES
RTM_WAT_DEFINE_CLASSES
RTM_OTH_DEFINE_CLASSES
RTM_ATM_DEFINE_CLASSES
RTM_TOP_DEFINE_CLASSES
```

```
; Compile HYD sources and define structures/classes

.run HYD_Tool.pro
.run HYD_Access.pro
.run HYD_Process.pro
.run HYD_Display.pro
.run HYD_Methods.pro

HYD_AC_DEFINE_STRUCT
HYD_DEFINE_CLASSES

; Get Context
st = HYD_TL_GET_CONTEXT(Ctx)
```

# Conclusions

RTM-IDL 1.0 package is part of a more general activity that aims at generating a dataset of simulated radiances at TOA and water leaving reflectances for the atmospheric correction and aerosol properties retrieval. The FEM code has been extensively exploited, while Hydrolight used as a reference for particular cases. The approach described in the document has shown several advantages:

- Unified definition of the coupled water-atmosphere system makes particularly easy the comparison of test cases to be solved with different RT models.
- Interface of the RT codes through IDL allows the writing of routines at higher level to perform systematic runs, to load and analyse the results taking advantage of all well-known IDL features.
- The addition of other models/components (other parameterisations for the definition of sea water IOPs, other aerosol models) is easily performed.

Some further activity is foreseen to make more convenient the use of RTM-IDL 1.0, in particular a widget-based GUI can be written to deal with the structure `SYS_IN` that defines the `SYSTEM`, as described in Section 1.5.



# Acknowledgements

The author would like to thank Barbara Bulgarelli for allowing the use of FEM and for her support in the understanding of the code and Frédéric Mélin for his precious suggestions for improvements of the RTM-IDL 1.0.

## Chapter 4

### Annexes

## 4.1 List of SYS-IDL routines

SYS-IDL routine list and description

This page was created by the IDL library routine `mk_html_help`. For more information on this routine, refer to the IDL Online Help Navigator or type:

```
? mk_html_help
```

at the IDL command line prompt.

Last modified: Mon Jan 31 17:21:45 2005.

---

### List of Routines

- \* RTM\_TL\_READ\_KEYWORD
- \* RTM\_TL\_PRINT\_ERROR
- \* RTM\_TL\_OBJ\_GET\_CLASS\_PAR
- \* RTM\_TL\_OBJ\_GET\_ARGUMENT
- \* RTM\_DD\_WAT\_COEFF
- \* RTM\_DD\_RADIANCE
- \* RTM\_DD\_IRRADIANCE
- \* RTM\_TOP\_DEFINE\_CLASSES
- \* RTM\_GEN\_DEFINE\_CLASSES
- \* VPROFILE::INITIAL
- \* VPROFILE::GETVALUE
- \* VPROFILE::PLOT
- \* GAUSSIAN::INITIAL
- \* GAUSSIAN::GETVALUE
- \* GAUSSIAN::PLOT
- \* UNIFORM::INITIAL
- \* UNIFORM::GETVALUE
- \* SERIE::READ
- \* SERIE::INITIAL
- \* SERIE::GETVALUE
- \* SERIE::GETAVGVALUE
- \* PHF::INITIAL
- \* PHF\_FILE::INITIAL
- \* PHF\_RAYL::INITIAL
- \* PHF\_TTHG::INITIAL
- \* RTM\_ATM\_DEFINE\_CLASSES
- \* COMP\_O3::INITIAL
- \* COMP\_AER::INITIAL
- \* COMP\_AER\_OPAC::INITIAL
- \* COMP\_MOL::INITIAL
- \* COMP\_O2::INITIAL

```

* ATM::INITIAL
* RTM_WAT_DEFINE_CLASSES
* ISRC::INITIAL
* A_MOD::INITIAL
* A_CONST::INITIAL
* A_USR::INITIAL
* A_PEF::INITIAL
* A_SEB::INITIAL
* A_PSM::INITIAL
* A_BRI::INITIAL
* A_CASE1::INITIAL
* A_CHLA1::INITIAL
* A_EXP::INITIAL
* A_CDOM::INITIAL
* RTM_OTH_DEFINE_CLASSES

```

-----

## Routine Descriptions

### RTM\_TL\_READ\_KEYWORD

[Next Routine] [List of Routines]

#### NAME:

RTM\_TL\_READ\_KEYWORD

#### PURPOSE:

This function parses a string array (str\_array) looking for the definition of a keyword, and returns the value defined, as a string (str\_read)

The string array SHALL be defined as following:

```

' keyword1  =  string1 '
' keyword2  =  string2 '

' keywordn  =  stringn '

```

IMPORTANT : str\_array SHALL NOT CONTAIN ';' char (otherwise disregarded)

#### CATEGORY:

RTM Tool

#### CALLING SEQUENCE:

Result = RTM\_TL\_READ\_KEYWORD(str\_array,keyword,str\_read)

INPUTS:  
 str\_array : array of strings to be parsed  
 keyword : keyword searched

KEYWORD PARAMETERS:

none

OUTPUTS:

str\_read : value

RETURN : 0 - Keyword found  
 >0 - Keyword not found

MODIFICATION HISTORY:

Written by: Marco Clerici, 24.05.04

(See RTM\_Tool.pro)

-----

RTM\_TL\_PRINT\_ERROR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

RTM\_TL\_PRINT\_ERROR

PURPOSE:

Print an error message on the standard error output

CATEGORY:

Tool

CALLING SEQUENCE:

RTM\_TL\_PRINT\_ERROR , Ctx , routine, level, message [,/NODATE] [,/INFO]  
 [,OPT\_LINE=opt\_line]

INPUTS:

Ctx : General Context

routine : Name of the calling routine

level : Severity level of the message : 0 - FATAL  
 1 - ERROR  
 2 - WARNING  
 3 - INFO

message : Error message text

KEYWORD PARAMETERS:

INFO : Flag to force the printing of the message, disregarding

the actual Ctx.ErrorLevel value

NODATE : Do not print the date and time

OPT\_LINE: Optional line, used to make the message more detailed

OUTPUTS:

RESTRICTIONS:

MODIFICATION HISTORY:

Written by: Marco Clerici, 24.05.04

(See RTM\_Tool.pro)

-----

RTM\_TL\_OBJ\_GET\_CLASS\_PAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

RTM\_TL\_OBJ\_GET\_CLASS\_PAR

PURPOSE:

Extract from a string the class name and initialisation parameters.

Syntax is:

str = '[ CLASS\_NAME: param1=val1, param2=val2 ...]'

The remaining part of the string is returned

CATEGORY:

RTM Tool

CALLING SEQUENCE:

Result = RTM\_TL\_OBJ\_GET\_CLASS\_PAR(str, class, param [,CTX=ctx])

INPUTS:

str : string in format described above

KEYWORD PARAMETERS:

Ctx : general context

OUTPUTS:

class : class name

param : parameter list

RETURN : 0 - Success  
          >0 - Failure

MODIFICATION HISTORY:

Written by: Marco Clerici, 24.05.04

(See RTM\_Tool.pro)

-----

RTM\_TL\_OBJ\_GET\_ARGUMENT

[Previous Routine] [Next Routine] [List of Routines]

NAME:

RTM\_TL\_OBJ\_GET\_ARGUMENT

PURPOSE:

Extract from a string array the string associated to the given name.

Args is like: args(n,2)

args(0,0)	= name1	args(0,1)	= value1
args(1,0)	= name2	args(1,1)	= value2
args(2,0)	= name3	args(2,1)	= value3

If the name is not found, empty string is returned

CATEGORY:

RTM Tool

CALLING SEQUENCE:

Result = RTM\_TL\_OBJ\_GET\_ARGUMENT(args,name,value)

INPUTS:

args : string array in format described above  
name : argument name

KEYWORD PARAMETERS:

none

OUTPUTS:

value : argument value

RETURN : 0 - Success  
          >0 - Failure

MODIFICATION HISTORY:

Written by: Marco Clerici, 17.05.04

(See RTM\_Tool.pro)

-----  
RTM\_DD\_WAT\_COEFF

[Previous Routine] [Next Routine] [List of Routines]

NAME:

RTM\_DD\_WAT\_COEFF

PURPOSE:

Display an wat coefficient (a,b,c or Kd), with different options:

1. Plot as 3D vs. depth/wavelength (keyword SURF=1)
2. Plot as 2D vs. depth, 1 curve for each wave (only total value)
3. Plot as 2D vs. depth, 1 curve for components, at 1 wave  
(keyword COMP=1 and iWAVE=i)

CATEGORY:

HYD Access

CALLING SEQUENCE:

RTM\_DD\_WAT\_COEFF, wat, ...

INPUTS:

wat : wat coefficient [depth,wave] or [depth,wave,comp]  
depths : depth array  
wave : wavelength array

KEYWORD PARAMETERS:

PS : print to Postscript file  
EPS : print to Encapsulated Postscript file  
NAME : wat coefficient label (for title/z-axis)  
UNIT : wat unit  
SURF : display in 3D  
OPTIC : optical depth (default is geometrical)  
INS\_COL : instrument specific colors (refer to wave array;  
default is SEAWIFS); If -1, plot in black & white  
COMP : if set, plot for each component. In this case, wat  
is [depth,wave,comp] and iWAVE should be defined  
When set, also defines the number of components  
to be plotted

OUTPUTS:

none



## MODIFICATION HISTORY:

Written by: Marco Clerici, 21.05.04

adapted from Hydrolight 4.1 IDL routine: ugfig6.pro

(See RTM\_Display.pro)

-----

## RTM\_DD\_RADIANCE

[Previous Routine] [Next Routine] [List of Routines]

### NAME:

RTM\_DD\_RADIANCE

### PURPOSE:

Display a radiance field as function of viewing dir and wave.

Note on view angle : the Mobley std is described in Light & Water \$11.1 (p.506)

There the xrange 0-180 -> azi=180, zen=0-180

(zen=0 means looking dnw, photons upw)

xrange 180-0 -> azi=0, zen=180-0

Here we join the two ranges into 1 range 0 - 180 - 0

Both Hydrolight and FEM modules must prepare the rad array joining results of two azimuth angles (e.g. azi=0 and azi=180 for principal plane)

### CATEGORY:

HYD Access

### CALLING SEQUENCE:

RTM\_DD\_RADIANCE, rad, ..

### INPUTS:

rad : radiance [wave,zen]

wave : wavelength array

view : zenithal viewing angle (in degrees)

### KEYWORD PARAMETERS:

PS : print to Postscript file

EPS : print to Encapsulated Postscript file

NAME : radiance name

TITLE : Plot title  
UNIT : radiance unit  
SURF : display in 3D  
INS\_COL : instrument specific colors (refer to wave array;  
default is SEAWIFS); If -1, plot in black & white  
LOG : display in logarithmic scale  
WNBR : define the window number which to plot to  
THETA : viewing zenith angles to highlight  
LYR : layer number (for title and .eps filename)  
RID : run Id (for .eps filename)

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 21.05.04  
adapted from Hydrolight 4.1 IDL routine: ugfig7.pro

(See RTM\_Display.pro)

-----

RTM\_DD\_IRRADIANT

[Previous Routine] [Next Routine] [List of Routines]

NAME:

RTM\_DD\_IRRADIANT

PURPOSE:

Display an irradiance field:

1. Plot as 3D vs. depth/wavelength (keyword SURF=1)
2. Plot as 2D vs. depth, 1 curve for each wave (only total value)

CATEGORY:

HYD Access

CALLING SEQUENCE:

RTM\_DD\_IRRADIANT, irr, ...

INPUTS:

irr : irradiance array [depth,wave]  
depths : depth array  
wave : wavelength array

## KEYWORD PARAMETERS:

PS : print to Postscript file  
EPS : print to Encapsulated Postscript file  
NAME : irradiance label (for z-axis)  
TITLE : display title  
UNIT : irradiance unit  
SURF : display in 3D  
OPTIC : optical depth (default is geometrical)  
INS\_COL : instrument specific colors (refer to wave array;  
default is SEAWIFS); If -1, plot in black & white  
LOG : display in logarithmic scale  
YLOG : use logarithmic y-axis  
OVERP : overplot (no for SURFACE). It is also used for offset in  
legend (set to 1,2,3 for multiple overplot)  
COLOR : force the use of a color. It is used only for overplot,  
and is intended for 1 wavelength. (no for SURFACE)  
LEGEND : token to print as a legend. (no for SURFACE)

## OUTPUTS:

none

## MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

(See RTM\_Display.pro)

-----

## RTM\_TOP\_DEFINE\_CLASSES

[Previous Routine] [Next Routine] [List of Routines]

### NAME:

RTM\_TOP\_DEFINE\_CLASSES

### PURPOSE:

Define the top level class

### CATEGORY:

RTM Tool

### CALLING SEQUENCE:

RTM\_TOP\_DEFINE\_CLASSES

### INPUTS:

none

KEYWORD PARAMETERS:

none

OUTPUTS:

sys\_in : {SYS\_IN} variable initialised

MODIFICATION HISTORY:

Written by: Marco Clerici, 21.05.04

(See RTM\_TOP\_define.pro)

-----

RTM\_GEN\_DEFINE\_CLASSES

[Previous Routine] [Next Routine] [List of Routines]

NAME:

RTM\_GEN\_DEFINE\_CLASSES

PURPOSE:

Define the following basic classes:

VPROFILE      <- GAUSSIAN  
                 <- UNIFORM  
                 <- SERIE

PHF            <- PHF\_FILE  
                 <- PHF\_RAYL  
                 <- PHF\_TTHG  
                 <- PHF\_RATIO  
                 <- PHF\_HYSCA  
                 <- PHF\_CDOM

COMP

CATEGORY:

RTM Tool

CALLING SEQUENCE:

RTM\_GEN\_DEFINE\_CLASSES

INPUTS:

none

KEYWORD PARAMETERS:

none

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 14.05.04

(See RTM\_GEN\_define.pro)

---

## VPROFILE::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

```
VPROFILE::Initial
```

PURPOSE:

```
Initialise a VPROFILE object
```

CALLING SEQUENCE:

```
VPROFILE::Initial [,TIT=tit] [,XNAME=xname] [,YNAME=yname] [,STAT=stat] $
                  [,Xv=xv]      [,Yv=yv]      [,FILE=file]
```

INPUT:

NONE

KEYWORD:

TIT: plot title

XNAME: x-axis title

YNAME: Y-axis title

STAT: object status

XV: x-values array (n.u.)

YV: y-values array (n.u.)

FILE: filename

ROUTINE:routine name

(See RTM\_GEN\_define.pro)

---

## VPROFILE::GETVALUE

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

VPROFILE::GetValue

PURPOSE:

Get the y value associated to a given x

CALLING SEQUENCE:

VPROFILE::GetValue, x, y

INPUT:

x: x value

OUTPUT:

y: y value

KEYWORD:

none

(See RTM\_GEN\_define.pro)

-----

VPROFILE::PLOT

[Previous Routine] [Next Routine] [List of Routines]

NAME:

VPROFILE::Plot

PURPOSE:

Plot the vertical profile

CALLING SEQUENCE:

VPROFILE::Plot [,XR=xr] [,YR=yr] [,COL=col]

INPUT:

none

OUTPUT:

none

KEYWORD:

XR: x-axis range

YR: y-axis range

COL: plot color

(See RTM\_GEN\_define.pro)

-----

GAUSSIAN::INITIAL



(See RTM\_GEN\_define.pro)

-----

## GAUSSIAN::PLOT

[Previous Routine] [Next Routine] [List of Routines]

### NAME:

GAUSSIAN::Plot

### PURPOSE:

Plot the vertical profile

### CALLING SEQUENCE:

GAUSSIAN::Plot [,XR=xr] [,YR=yr] [,XSTEP=xstep] [,COL=col]

### INPUT:

none

### OUTPUT:

none

### KEYWORD:

XR: x-axis range

YR: y-axis range

YR: y-axis range

XSTEP: x-axis step

(See RTM\_GEN\_define.pro)

-----

## UNIFORM::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

### NAME:

UNIFORM::Initial

### PURPOSE:

Initialise a UNIFORM object

### CALLING SEQUENCE:

UNIFORM::Initial [,TIT=tit] [,XNAME=xname] [,YNAME=yname] [,STAT=stat] \$  
[,C=c]

### INPUT:

NONE



KEYWORD:

some keywords as for VPROFILE::Initial, and additional:

C:        profile constant value

(See RTM\_GEN\_define.pro)

-----

UNIFORM::GETVALUE

[Previous Routine] [Next Routine] [List of Routines]

NAME:

UNIFORM::GetValue

PURPOSE:

Get the y value associated to a given x

CALLING SEQUENCE:

UNIFORM::GetValue, x, y

INPUT:

x:    x value

OUTPUT:

y:    y value

KEYWORD:

none

(See RTM\_GEN\_define.pro)

-----

SERIE::READ

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SERIE::Read

PURPOSE:

Read a 'serie' profile from file

CALLING SEQUENCE:

SERIE::Read, filename [,COL=col]

INPUT:  
    filename: name of the file to be read

OUTPUT:  
    none

KEYWORD:  
    COL: column hosting y-values (default is 1)  
        x-value always is col 1

(See RTM\_GEN\_define.pro)

---

SERIE::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:  
    SERIE::Initial

PURPOSE:  
    Initialise a SERIE object

CALLING SEQUENCE:  
    SERIE::Initial, [,TIT=tit] [,XNAME=xname] [,YNAME=yname] [,STAT=stat] \$  
                    [,FILE=file][,COL=col]

INPUT:  
    NONE

KEYWORD:  
    some keywords as for VPROFILE::Initial, and additional:  
    FILE: filename  
    COL: column hosting y-values (default is 1)

(See RTM\_GEN\_define.pro)

---

SERIE::GETVALUE

[Previous Routine] [Next Routine] [List of Routines]

NAME:  
    SERIE::GetValue

PURPOSE:

Get the y value associated to a given x

CALLING SEQUENCE:

SERIE::GetValue, x, y

INPUT:

x: x value

OUTPUT:

y: y value

KEYWORD:

none

(See RTM\_GEN\_define.pro)

-----

SERIE::GETAVGVALUE

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SERIE::GetAvgValue

PURPOSE:

Given two x-values, return the y average

CALLING SEQUENCE:

SERIE::GetAvgValue, x1, x2, y

INPUT:

x1: first x value

x2: second x value

OUTPUT:

y: y value

KEYWORD:

LOG: compute logarithmic average

(See RTM\_GEN\_define.pro)

-----

PHF::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

PHF::Initial

PURPOSE:

Initialise a PHF object

CALLING SEQUENCE:

PHF::Initial [,ID=id]

INPUT:

none

KEYWORD:

ID: phase function identifier

(See RTM\_GEN\_define.pro)

-----

PHF\_FILE::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

PHF\_FILE::Initial

PURPOSE:

Initialise a PHF\_FILE object

CALLING SEQUENCE:

PHF\_FILE::Initial [,ID=id] [,TYPE=type] [,NAME=name]

INPUT:

none

KEYWORD:

ID: phase function identifier

TYPE: phf type

FILE: phf filename

(See RTM\_GEN\_define.pro)

-----

PHF\_RAYL::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

PHF\_RAYL::Initial

PURPOSE:

Initialise a PHF\_RAYL object

CALLING SEQUENCE:

PHF\_RAYL::Initial [,ID=id] [,XCOF1=xcof1] [,XCOF1=xcof1]

INPUT:

none

KEYWORD:

ID: phase function identifier

XCOF1: first Legendre coefficient

XCOF2: third Legendre coefficient

(See RTM\_GEN\_define.pro)

-----

PHF\_TTHG::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

PHF\_TTHG::Initial

PURPOSE:

Initialise a PHF\_TTHG object

CALLING SEQUENCE:

PHF\_TTHG::Initial [,ID=id] [G1=g1] [,G2=g2] [,AS=as]

INPUT:

none

KEYWORD:

ID: phase function identifier

G1: TTHG coefficient g1

G2: TTHG coefficient g2

AS: TTHG coefficient as

(See RTM\_GEN\_define.pro)

-----

## RTM\_ATM\_DEFINE\_CLASSES

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

### NAME:

RTM\_ATM\_DEFINE\_CLASSES

### PURPOSE:

Define the classes related to atmosphere

### CATEGORY:

RTM ATM

### CALLING SEQUENCE:

RTM\_ATM\_DEFINE\_CLASSES

### INPUTS:

none

### KEYWORD PARAMETERS:

none

### OUTPUTS:

none

### MODIFICATION HISTORY:

Written by: Marco Clerici, 14.05.04

(See RTM\_ATM\_define.pro)

-----

## COMP\_03::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

### NAME:

COMP\_03::Initial

### PURPOSE:

Initialise a COMP\_03 object

### CALLING SEQUENCE:

COMP\_03::Initial, args [,ATM03=Atm03]

### INPUT:

args: string array[2,n], e.g. : [ id, , '03std']

```
[ prof, [SERIE: file="Ozone_U76.dat"] ]  
[ 03 , 333.00 ]
```

KEYWORD:

ATM03: ozone total column [DBU], used if args.03 undef

(See RTM\_ATM\_define.pro)

---

COMP\_AER::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_AER::Initial

PURPOSE:

Initialise a COMP\_AER object

CALLING SEQUENCE:

COMP\_AER::Initial, args

INPUT:

args: string array[2,n], e.g. : [ a , 0.05 ]  
[ nu , 1.5 ]

[ PHF , [PHF\_TTHG: as=0., g1=0., g2=0.]]

KEYWORD:

none

(See RTM\_ATM\_define.pro)

---

COMP\_AER\_OPAC::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_AER\_OPAC::Initial

PURPOSE:

Initialise a COMP\_AER\_OPAC object

CALLING SEQUENCE:

COMP\_AER\_OPAC::Initial, args

INPUT:  
args: string array[2,n], e.g. : [ aer\_type, 'MARPL' ]  
[ rh , 95. ]  
[ vls , 'OPAC\_01' ]  
[ a865 , 0.05 ]

KEYWORD:  
none

(See RTM\_ATM\_define.pro)

-----

COMP\_MOL::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:  
COMP\_MOL::Initial

PURPOSE:  
Initialise a COMP\_MOL object

CALLING SEQUENCE:  
COMP\_MOL::Initial, args [,SurfPres=SurfPres]

INPUT:  
args: string array[2,n], e.g. : [ id , 'mol' ]  
[ PHF , [PHF\_TTHG: as=0., g1=0., g2=0.]]  
[ p0 , 1013.25 ]

KEYWORD:  
SurfPres: surface pressure, used if p0 is unset

(See RTM\_ATM\_define.pro)

-----

COMP\_O2::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:  
COMP\_O2::Initial

PURPOSE:  
Initialise a COMP\_O2 object

CALLING SEQUENCE:



COMP\_02::Initial, args [,SurfPres=SurfPres]

INPUT:

args: string array[2,n], e.g. : [ id , '02std']

[ prof, [SERIE: file="Ozone\_U76.dat"] ]  
[ p0 , 1013,25]

KEYWORD:

SurfPres: surface pressure, used if p0 is unset

(See RTM\_ATM\_define.pro)

-----

ATM::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

ATM::Initial

PURPOSE:

Initialise an ATM object

CALLING SEQUENCE:

ATM::Initial, atm\_in [,HYDR=hydr]

INPUT:

atm\_in: structure defined in RTM\_ATM\_DEFINE\_CLASSES

KEYWORD:

HYDR: initialisation for Hydrolight

(See RTM\_ATM\_define.pro)

-----

RTM\_WAT\_DEFINE\_CLASSES

[Previous Routine] [Next Routine] [List of Routines]

NAME:

RTM\_WAT\_DEFINE\_CLASSES

PURPOSE:

Define the classes related to water

CATEGORY:  
RTM Tool

CALLING SEQUENCE:

RTM\_WAT\_DEFINE\_CLASSES

INPUTS:

none

KEYWORD PARAMETERS:

none

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 14.05.04

(See RTM\_WAT\_define.pro)

-----

ISRC::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

ISRC::Initial

PURPOSE:

Initialise a ISRC object (internal sources - only for Hydrolight)

CALLING SEQUENCE:

ISRC::Initial [,BIOLUM=biolum] [,CHLFLU=CHLFLU] [,CDOMFLU=cdomflu] \$  
[,RAMAN=raman] [,COMPCHLA=compchla]

INPUT:

none

KEYWORD:

BIOLUM: bioluminescence

CHLFLU: chlorophyll luminescence

CDOMFLU: cdom luminescence

RAMAN: Raman Scattering

COMPCHLA: chlorophyll component used for chlorophyll luminescence

(See RTM\_WAT\_define.pro)

-----

A\_MOD::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

A\_MOD::Initial

PURPOSE:

Initialise a A\_MOD object

CALLING SEQUENCE:

A\_MOD::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

FILE: a\_mod file name

(See RTM\_WAT\_define.pro)

-----

A\_CONST::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

A\_CONST::Initial

PURPOSE:

Initialise a A\_CONST object (set a\_coeff to a constant value)

CALLING SEQUENCE:

A\_CONST::Initial [,ID=id] [,VAL=val]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

VAL: a\_coeff constant value

(See RTM\_WAT\_define.pro)

-----

A\_USR::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

A\_USR::Initial

PURPOSE:

Initialise a A\_USR object (a\_coeff read from a file)

CALLING SEQUENCE:

A\_CONST::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

FILE: file containing a\_coeff vs. wl

(See RTM\_WAT\_define.pro)

-----

A\_PeF::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

A\_PeF::Initial

PURPOSE:

Initialise a A\_PeF object (Pope & Fry model)

CALLING SEQUENCE:

A\_PeF::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

FILE: file containing a\_coeff vs. wl for Pope & Fry

(See RTM\_WAT\_define.pro)

-----

A\_SEB::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

A\_SeB::Initial

PURPOSE:

Initialise a A\_SeB object (Smith & Baker model)

CALLING SEQUENCE:

A\_SeB::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

FILE: file containing a\_coeff vs. wl for Smith & Baker

(See RTM\_WAT\_define.pro)

-----

A\_PSM::INITIAL

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

A\_PSM::Initial

PURPOSE:

Initialise a A\_PSM object (Priour/Sathyenranath/Morel model)

CALLING SEQUENCE:

A\_PSM::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

FILE: file containing a\_coeff vs. wl for Priour/Sathyenranath/Morel

(See RTM\_WAT\_define.pro)

-----

A\_BRI::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_BRI::Initial

PURPOSE:

Initialise a A\_BRI object (Bricaud model)

CALLING SEQUENCE:

A\_BRI::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

FILE: file containing a\_coeff vs. wl for Bricaud

(See RTM\_WAT\_define.pro)

-----

A\_CASE1::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CASE1::Initial

PURPOSE:

Initialise a A\_CASE1 object ('CASE1 model')

This model take into account Chla Absorbtion as from  
Bricaud 95 and adds CDOM/MIN cohvarying contributions

CALLING SEQUENCE:

A\_CASE1::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

FILE: file containing a\_coeff vs. wl for CASE1

(See RTM\_WAT\_define.pro)

-----

A\_CHLA1::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CHLA1::Initial

PURPOSE:

Initialise a CHLA1 object ('CHLA1 model')  
This model take into account Chla Absorbtion as from  
Bricaud 95 and adds MIN absorbtion. It is conceived to  
be used with case2 water

CALLING SEQUENCE:

CHLA1::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier  
FILE: file containing a\_coeff vs. wl for CHLA1

(See RTM\_WAT\_define.pro)

-----

A\_EXP::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_EXP::Initial

PURPOSE:

Initialise a A\_EXP object (only for Hydrolight)

CALLING SEQUENCE:

A\_EXP::Initial [,ID=id] [,FILE=file] [,WREF=wref] [,AREF=aref] [,GAMM=gamma]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier  
FILE: file containing a\_coeff vs. wl for CHLA1

WREF: reference wavelength  
AREF: a coefficient at the reference wavelength  
GAMM: exponential decay constant

(See RTM\_WAT\_define.pro)

---

A\_CDOM::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CDOM::Initial

PURPOSE:

Initialise a A\_CDOM object

CALLING SEQUENCE:

A\_CDOM::Initial [,ID=id] [,SYS=SYS]

INPUT:

none

KEYWORD:

ID: a\_mod function identifier

SYS: constant exponential decay

(See RTM\_WAT\_define.pro)

---

RTM\_OTH\_DEFINE\_CLASSES

[Previous Routine] [List of Routines]

NAME:

RTM\_OTH\_DEFINE\_CLASSES

PURPOSE:

Define the following classes:

SUN

LOCAT

GRID

WAVE

OLS

BOTTOM



CATEGORY:

RTM Tool

CALLING SEQUENCE:

RTM\_OTH\_DEFINE\_CLASSES

INPUTS:

none

KEYWORD PARAMETERS:

none

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 14.05.04

(See RTM\_OTH\_define.pro)

-----

## 4.2 List of FEM-IDL routines

FEM-IDL routines list and description

This page was created by the IDL library routine `mk_html_help`. For more information on this routine, refer to the IDL Online Help Navigator or type:

`? mk_html_help`

at the IDL command line prompt.

Last modified: Mon Jan 31 17:21:12 2005.

---

### List of Routines

- \* FEM\_AC\_DEFINE\_STRUCT
- \* FEM\_AC\_INIT\_STR\_IN1
- \* FEM\_AC\_INIT\_STR\_IN2
- \* FEM\_AC\_WRITE\_MAIN
- \* FEM\_AC\_WRITE\_INPUT1
- \* FEM\_AC\_WRITE\_INPUT2
- \* FEM\_AC\_WRITE\_MOM
- \* FEM\_AC\_READ\_OUT1
- \* FEM\_AC\_READ\_OUT2
- \* FEM\_AC\_READ\_OUT3
- \* FEM\_AC\_READ\_INV
- \* FEM\_DD\_ANL\_RUN
- \* FEM\_DD\_RADIANCE
- \* FEM\_DD\_REFLECT
- \* FEM\_DD\_IRRADIANT
- \* FEM\_DD\_INPUTS
- \* FEM\_PP\_RUN
- \* FEM\_PP\_LOOP
- \* FEM\_PP\_READ
- \* FEM\_PP\_SAVE
- \* FEM\_PP\_OUT3\_DIFF
- \* FEM\_PP\_INV\_DIFF
- \* FEM\_TL\_GET\_CONTEXT
- \* FEM\_DEFINE\_CLASSES
- \* PHF::FEMTRUNC
- \* PHF\_FILE::FEMGET\_XCOF
- \* PHF\_TTHG::FEMGET\_XCOF
- \* PHF\_RAYL::FEMGET\_XCOF
- \* A\_CONST::FEMGET\_A
- \* A\_PEF::FEMGET\_A
- \* A\_BRI::FEMGET\_A

```

* A_CDOM::FEMGET_A
* A_CHLA1::FEMGET_A
* A_CASE1::FEMGET_A
* B_CONST::FEMGET_B
* B_SEB::FEMGET_B
* B_POW::FEMGET_B
* B_MEM::FEMGET_B
* B_MIN::FEMGET_B
* COMP::FEMGETVAR
* COMP_PW::FEMGETVAR
* COMP_CHLA::FEMGETVAR
* COMP_CDOM::FEMGETVAR
* COMP_MIN::FEMGETVAR
* WAT::FEMGETVAR
* COMP_03::FEMGETVAR
* COMP_03::FEMGETVAR_VALID
* COMP_AER::FEMGETVAR
* COMP_AER_OPAC::FEMGETVAR
* COMP_AER_OPAC::FEMGETVARBIN
* COMP_MOL::FEMGETVAR
* COMP_02::FEMGETVAR
* ATM::FEMGETVAR
* SYSTEM::FEM_INPUT
* SYSTEM::FEM_OPTPRO
* SYSTEM::TEST_OPAC

```

-----

## Routine Descriptions

### FEM\_AC\_DEFINE\_STRUCT

[Next Routine] [List of Routines]

#### NAME:

FEM\_AC\_DEFINE\_STRUCT

#### PURPOSE:

Define structures fo I/O

#### CATEGORY:

FEM Access

#### CALLING SEQUENCE:

Result = FEM\_AC\_DEFINE\_STRUCT([/OLDER\_T004])

#### INPUTS:

none  
KEYWORD PARAMETERS:  
 OLDER\_T004: if set, a STR\_INV structure suitable for  
 FEM outputs till simulation T003 is created  
OUTPUTS:  
 none  
RETURN : 0 - OK  
 >0 - Error  
  
MODIFICATION HISTORY:  
 Written by: Marco Clerici, 21.04.04  
  
(See FEM\_Access.pro)

---

FEM\_AC\_INIT\_STR\_IN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:  
 FEM\_AC\_INIT\_STR\_IN1

PURPOSE:  
 Initialise structure STR\_INPUT1

CATEGORY:  
 FEM Access

CALLING SEQUENCE:

Result = FEM\_AC\_INIT\_STR\_IN1(str\_in1)

INPUTS:  
 str\_in1 : structure for input1 file

KEYWORD PARAMETERS:  
 none  
OUTPUTS:  
 none  
RETURN : 0 - OK  
 >0 - Error

MODIFICATION HISTORY:  
 Written by: Marco Clerici, 26.05.04

(See FEM\_Access.pro)

---

FEM\_AC\_INIT\_STR\_IN2

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

FEM\_AC\_INIT\_STR\_IN2

PURPOSE:

Initialise structure STR\_INPUT2

CATEGORY:

FEM Access

CALLING SEQUENCE:

Result = FEM\_AC\_INIT\_STR\_IN2(str\_in2)

INPUTS:

str\_in2 : structure for input2 file

KEYWORD PARAMETERS:

none

OUTPUTS:

none

RETURN : 0 - OK  
          >0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

(See FEM\_Access.pro)

---

FEM\_AC\_WRITE\_MAIN

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

FEM\_AC\_WRITE\_MAIN

PURPOSE:

Write the FEM input file containing the name of all the I/O files

CATEGORY:

FEM Access

CALLING SEQUENCE:

```
Result = FEM_AC_WRITE_MAIN(Ctx [,DIR=dir])
```

INPUTS:

Ctx : global context

KEYWORD PARAMETERS:

dir : main.dat destination dir - if undef Ctx.ExecDir is used

OUTPUTS:

none

RETURN : 0 - OK  
>0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

(See FEM\_Access.pro)

-----

FEM\_AC\_WRITE\_INPUT1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_AC\_WRITE\_INPUT1

PURPOSE:

Write the FEM input file 1

CATEGORY:

FEM Access

CALLING SEQUENCE:

```
Result = FEM_AC_WRITE_INPUT1(Ctx,Str_in1,...)
```

INPUTS:

Ctx : global context  
Str\_in1 : structure containing data to be written

KEYWORD PARAMETERS:

DIR : input1 file directory

FILE : input name 1 as define for FEM in main.dat

OUTPUTS:

none

RETURN : 0 - OK  
>0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

(See FEM\_Access.pro)

-----

FEM\_AC\_WRITE\_INPUT2

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_AC\_WRITE\_INPUT2

PURPOSE:

Write the FEM input file 2

CATEGORY:

FEM Access

CALLING SEQUENCE:

Result = FEM\_AC\_WRITE\_INPUT2(Ctx, str\_in2 [,DIR=dir] [,FILE=file])

INPUTS:

Ctx : global context  
str\_in2 : structure containing data to be written

KEYWORD PARAMETERS:

DIR : input1 file directory  
FILE : input name 2 as define for FEM in main.dat

OUTPUTS:

none

RETURN : 0 - OK  
>0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 27.05.04

(See FEM\_Access.pro)

-----

## FEM\_AC\_WRITE\_MOM

[Previous Routine] [Next Routine] [List of Routines]

### NAME:

FEM\_AC\_WRITE\_MOM

### PURPOSE:

Write the FEM PHF Legendre Coefficient file

### CATEGORY:

FEM Access

### CALLING SEQUENCE:

Result = FEM\_AC\_WRITE\_MOM(Ctx, Str\_mom, [,DIR=dir] [,FILE=file] [,BIN=bin])

### INPUTS:

Ctx : global context  
Str\_mom : structure containing xcof for each layer

### KEYWORD PARAMETERS:

DIR : input1 file directory  
FILE : input name 1 as define for FEM in main.dat  
BIN : write the file as F77\_UNFORMATTED (binary)

### OUTPUTS:

none  
RETURN : 0 - OK  
          >0 - Error

### MODIFICATION HISTORY:

Written by: Marco Clerici, 04.06.04

(See FEM\_Access.pro)

-----

## FEM\_AC\_READ\_OUT1

[Previous Routine] [Next Routine] [List of Routines]

### NAME:



FEM\_AC\_READ\_OUT1

PURPOSE:

Read output 1 file

CATEGORY:

FEM Access

CALLING SEQUENCE:

Result = FEM\_AC\_READ\_OUT1(Ctx,str\_out1 [,DIR=dir] [,FILE=file] )

INPUTS:

Ctx : global context

KEYWORD PARAMETERS:

file : output name 1 ( as define in main.dat)  
It is the full path

OUTPUTS:

Str\_out1 : structure to host read data

RETURN : 0 - OK  
>0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 27.05.04

(See FEM\_Access.pro)

-----

FEM\_AC\_READ\_OUT2

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_AC\_READ\_OUT2

PURPOSE:

Read output 2 file

CATEGORY:

FEM Access

CALLING SEQUENCE:

Result = FEM\_AC\_READ\_OUT2(Ctx,str\_out2 [,DIR=dir] [,FILE=file])

INPUTS:

Ctx : global context

KEYWORD PARAMETERS:

DIR : output direct; if not defined is taken from Ctx  
FILE : output name 2 as define for FEM in iolist.txt  
if not defined is taken from Ctx

OUTPUTS:

Str\_out2 : structure to host read data  
RETURN : 0 - OK  
>0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 27.05.04

(See FEM\_Access.pro)

-----  
FEM\_AC\_READ\_OUT3

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_AC\_READ\_OUT3

PURPOSE:

Read output 3 file (binary format)

CATEGORY:

FEM Access

CALLING SEQUENCE:

Result = FEM\_AC\_READ\_OUT3(Ctx, str\_out3 [,DIR=dir] [,FILE=file]  
[,OLD=old] [,RID=rid] [,/NODIR])

INPUTS:

Ctx : global context

KEYWORD PARAMETERS:

DIR : output direct; if not defined is taken from Ctx  
FILE : output name 3 (otherwise taken from Ctx)  
OLD : 'old' binary file format ( i.e. before 14.06.04,  
e.g. valid\_Mobley and valid\_07.06.04)  
RID : run id used to build up output name 3.

It overwrites FILE keyword!!

NODIR : if set, 'filename' is used as a fullpath, and no any dir  
is added (neither from keyword nor from default)

OUTPUTS:

Str\_out3 : structure to host read data

RETURN : 0 - OK  
>0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 27.05.04

(See FEM\_Access.pro)

---

## FEM\_AC\_READ\_INV

[Previous Routine] [Next Routine] [List of Routines]

### NAME:

FEM\_AC\_READ\_INV

### PURPOSE:

Read inversion binary file

### CATEGORY:

FEM Access

### CALLING SEQUENCE:

Result = FEM\_AC\_READ\_INV(Ctx, str\_inv [,DIR=dir] [,FILE=file]  
[,RID=rid] [,/OLDER\_T004])

### INPUTS:

Ctx : global context

### KEYWORD PARAMETERS:

DIR : output direct; if not defined is taken from Ctx

FILE : inversion file name

RID : run id used to build up filename  
It overwrite FILE keyword!!

OLDER\_T004 : dataset older than T004 -> some fields missing

### OUTPUTS:

Str\_inv : structure to host read data

RETURN : 0 - OK  
          >0 - Error

MODIFICATION HISTORY:

Written by: Marco Clerici, 17.06.04

(See FEM\_Access.pro)

-----

FEM\_DD\_ANL\_RUN

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_DD\_ANL\_RUN

PURPOSE:

Analyse the results of a run on 1 wave. It is intended to be called  
after FEM\_PP\_RUN

CATEGORY:

FEM Display

CALLING SEQUENCE:

FEM\_DD\_ANL\_RUN, runid, ..

INPUTS:

Ctx : Global context  
file : binary output file name (see keyword RUNID)

KEYWORD PARAMETERS:

RID : run identifier (if set, filename is rebuilt and 'file' arg  
overwritten)  
IRR\_UP : display irradiance upward  
IRR\_DW : display irradiance downward  
  
RADI : display radiance at the given level ( note: first level (TOA)  
here is 1, but is passed to FEM\_DD\_RADIANCE as 0)  
  
REFL : display reflectance at the given level (note: first level (TOA)  
here is 1, but is passed to FEM\_DD\_RADIANCE as 0)  
  
INPUT : display FEM inputs (tau-ssa)

DIR : directory containing results (if unset, Ctx.OutputDir is used)  
LOG : rad/irrad/inputs logarithmic axis  
EPS : print to eps file  
  
PHI : define radiance/reflectance plane (def.is phi=0 -> princ. plane)  
OLD : define an old binary output file format  
RAD\_3D : activate the rad\_3d GUI module

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 10.06.04

(See FEM\_Display.pro)

-----

FEM\_DD\_RADIANCE

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_DD\_RADIANCE

PURPOSE:

Display a radiance field as function of viewing dir

CATEGORY:

FEM Display

CALLING SEQUENCE:

FEM\_DD\_RADIANCE, str\_out, ..

INPUTS:

str\_out : structure containing FEM results: it can be either  
{STR\_OUTPUT3} or {STR\_INVERS} type

KEYWORD PARAMETERS:

DEPTH : index for the depth  
PHI : index for the azimuth (default is principal plane)  
LOG : set radiance log axis  
WNBR : define the window number which to plot to  
EPS : print to Encapsulated Postscript file  
PS : print to Postscript file

RID : run Id (for .eps filename)

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 27.05.04

(See FEM\_Display.pro)

-----

FEM\_DD\_REFLECT

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_DD\_REFLECT

PURPOSE:

Display a normalised reflectance field as function of viewing dir

Reflectance is computed as :

$$\text{Refl} = \text{Rad} / \text{Ed} * \cos(\text{SZA}) * !\pi \quad (\text{Sun-Earth distance correction missing})$$

CATEGORY:

FEM Display

CALLING SEQUENCE:

FEM\_DD\_REFLECT, str\_out, ..

INPUTS:

str\_out : structure containing FEM results

KEYWORD PARAMETERS:

DEPTH : index for the depth

PHI : index for the azimuth (default is principal plane)

LOG : set radiance log axis

EPS : print to Encapsulated Postscript file

PS : print to Postscript file

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 14.06.04

(See FEM\_Display.pro)

-----

FEM\_DD\_IRRADIAN

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_DD\_IRRADIAN

PURPOSE:

Display an irradiance field vs. optical depth (optionally vs.  
wave as well)

CATEGORY:

FEM Display

CALLING SEQUENCE:

FEM\_DD\_IRRADIAN, str\_out, ..

INPUTS:

str\_out : structure containing FEM results (str output 1)

KEYWORD PARAMETERS:

TYPE	: irradiance type	('Ed','Eu','Difd','Difu','Dird','Refu')
LOG	: use logarithmic x-axis	
YLOG	: use logarithmic y-axis	
SURF	: 3D plot (vs. depth/wave)	
OPTIC	: plot vs. optical depth	(always for the time being: internally forced)
WAVE	: wavelength (is not in the FEM output file)	- def. = 412
OVERP	: overplot. Not foreseen together with /SURF	
COLOR	: force the use of a color. It is used only for overplot,	and is intended for 1 wavelength.
XTITLE	: used to force xtitle (convenient for overplot cases)	
LEGEND	: print a legend (convenient for overplot cases)	
TITLE	: title	
EPS	: print to Encapsulated Postscript file	
PS	: print to Postscript file	

OUTPUTS:

none

#### MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

(See FEM\_Display.pro)

-----

#### FEM\_DD\_INPUTS

[Previous Routine] [Next Routine] [List of Routines]

#### NAME:

FEM\_DD\_INPUTS

#### PURPOSE:

Display a FEM inputs (ssa/tau) vs. depth (optical)

#### CATEGORY:

FEM Display

#### CALLING SEQUENCE:

FEM\_DD\_INPUTS, str\_out, ..

#### INPUTS:

str\_out : structure containing FEM results (str output 3)  
vls : given vls

#### KEYWORD PARAMETERS:

LOG : use logarithmic axis  
OPTIC : plot vs. optical depth (always, for the time being)  
WAVE : wavelength (is not in the FEM output file) - def. = 412  
OVERP : overplot  
COLOR : force the use of a color. It is used only for overplot,  
and is intended for 1 wavelength.  
XTITLE : used to force xtitle (convenient for overplot cases)  
LEGEND : print a legend (convenient for overplot cases) - TBD  
TITLE : title  
SSA : plot ssa (default is tau)  
EPS : print to Encapsulated Postscript file  
PS : print to Postscript file

#### OUTPUTS:

none



## MODIFICATION HISTORY:

Written by: Marco Clerici, 09.06.04

(See FEM\_Display.pro)

-----

## FEM\_PP\_RUN

[Previous Routine] [Next Routine] [List of Routines]

### NAME:

FEM\_PP\_RUN

### PURPOSE:

Write the input files and make the FEM run (for 1 wavelenght)

### CATEGORY:

FEM Access

### CALLING SEQUENCE:

Result = FEM\_PP\_RUN(Ctx,...)

### INPUTS:

Ctx : global context

Str\_in1 : Input1 structure (or undefined if file has not to be written)

Str\_in2 : Input2 structure (or undefined if file has not to be written)

Str\_mom : Legendre moments structure (or undefined if file has not to be written)

### KEYWORD PARAMETERS:

RID : run ID - if set, the output files name is modified  
used for valid\_07.06.04, could be replaced by FEM\_PP\_LOOP  
(many wavelenghts)

IN\_DIR : input directory ( if not set the one from Ctx is used)

OUT\_DIR : output directory ( if not set the one from Ctx is used)

SAVEIN : save the input files, by naming them according to RID

FREEMEM : do free str\_out after saving

Note : if IN/OUT dir are set, Ctx values are updated

OUTPUTS:

none  
RETURN : 0 - File written OK  
          >0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

19.07.04 : the executable name changes according to the nodename  
(UNIX machine), unless /NOLOCAL is set

(See FEM\_Process.pro)

-----  
FEM\_PP\_LOOP

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_PP\_LOOP

PURPOSE:

Run the FEM over several wavelenghts. A different set of output file  
(and input files, if SAVEIN set) is produced for each wavelenght, unless  
RID is unset.

CATEGORY:

FEM Process

CALLING SEQUENCE:

Result = FEM\_PP\_LOOP(Ctx,sys,...)

INPUTS:

Ctx : global context  
Sys : system object; it contains the wave description  
Vls : VLS to be used for run

KEYWORD PARAMETERS:

RID : run ID - if set, the output files name is modified -  
strongly recommended option  
IN\_DIR : input directory ( if not set the one from Ctx is used)  
OUT\_DIR : output directory ( if not set the one from Ctx is used)  
SAVE\_IN : if set, the input file names depend on the wavelength  
(so, they are not overwritten)

Note : if IN/OUT dir are set, Ctx values are updated

OUTPUTS:

none

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 09.06.04

(See FEM\_Process.pro)

-----

FEM\_PP\_READ

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_PP\_READ

PURPOSE:

Read the results of a FEM run

CATEGORY:

FEM Access

CALLING SEQUENCE:

Result = FEM\_PP\_READ(Ctx,...)

INPUTS:

Ctx : global context  
Str\_out1 : Output1 structure (used to read 'report' file)  
Str\_out2 : Output2 structure (used to read radiance from binary file)  
Str\_out4 : Output4 structure (used to read fluxes from binary file)

KEYWORD PARAMETERS:

RID : run ID - if set, the output files name is modified, e.g.  
output1name\_RAD.TXT -> output1name\_'RID'\_RAD.TXT  
DIR : output directory ( if not set the one from Ctx is used)

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

NOTE: output structures are generated by the calling routines. If an output  
does not have to be read, put 0 as argument instead of the structure

MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

(See FEM\_Process.pro)

-----

FEM\_PP\_SAVE

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_PP\_SAVE

PURPOSE:

Save quantities for 'inversion' from results of 1 FEM run

CATEGORY:

FEM Access

CALLING SEQUENCE:

Result = FEM\_PP\_SAVE(Ctx,...)

INPUTS:

Ctx : global context  
Str\_out : binary output file structure

KEYWORD PARAMETERS:

RID : run ID, which defines the output file name  
DIR : output directory ( if not set the one from Ctx is used)  
KEEPMEM : do not free str\_out after saving  
OLDER\_T004 : dataset older than T004 -> some fields missing

OUTPUTS:

none

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 15.06.04

(See FEM\_Process.pro)

-----

FEM\_PP\_OUT3\_DIFF

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_PP\_OUT3\_DIFF

PURPOSE:

Compute and report differences between two STR\_OUTPUT3

CATEGORY:

FEM Process

CALLING SEQUENCE:

Result = FEM\_PP\_OUT3\_DIFF(Ctx,str1,str2 [,WL=wl)

INPUTS:

Ctx : global context  
Str1 : binary output file structure  
Str2 : binary output file structure

KEYWORD PARAMETERS:

WL : compare only water leaving quantities  
NO\_DD\_TOA : do not consider in comparison downward  
diffuse irradi. at TOA

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 17.06.04

(See FEM\_Process.pro)

-----

## FEM\_PP\_INV\_DIFF

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

### NAME:

FEM\_PP\_INV\_DIFF

### PURPOSE:

Compute and report differences between two STR\_INVERS

### CATEGORY:

FEM Process

### CALLING SEQUENCE:

Result = FEM\_PP\_INV\_DIFF(Ctx,str1,str2 )

### INPUTS:

Ctx : global context  
Str1 : inversion structure  
Str2 : inversion structure

### KEYWORD PARAMETERS:

### OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

### MODIFICATION HISTORY:

Written by: Marco Clerici, 23.12.04

(See FEM\_Process.pro)

-----

## FEM\_TL\_GET\_CONTEXT

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

### NAME:

FEM\_TL\_GET\_CONTEXT

### PURPOSE:

This function creates the general context and initialises it.

The general context hosts the User Defined set-up Parameters and the

Global Constants used by FEM.

The User Defined set-up Parameters are initialised with default values that can be overwritten by the values defined in the Configuration File.

The Global Constants are hard-coded and SHOULD NOT be modified by the user.

CATEGORY:

FEM Tool

CALLING SEQUENCE:

Result = FEM\_TL\_GET\_CONTEXT(Ctx, [,DIR=dir] [,/PRINT])

INPUTS:

none

KEYWORD PARAMETERS:

DIR : Directory containing Configuration File  
PRINT : Print User Defined Parameters after reading

OUTPUTS:

Ctx : General FEM context

RETURN : 0 - Success  
>0 - Failure

MODIFICATION HISTORY:

Written by: Marco Clerici, 08.01.03

(See FEM\_Tool.pro)

-----

FEM\_DEFINE\_CLASSES

[Previous Routine] [Next Routine] [List of Routines]

NAME:

FEM\_DEFINE\_CLASSES

PURPOSE:

Define FEM specific classes, not defined at RTM (system) level:  
VLS : is the vertical layer structure used in computation  
GRID: computation discretisation

CATEGORY:  
FEM Methods

CALLING SEQUENCE:

FEM\_DEFINE\_CLASSES

INPUTS:

none

KEYWORD PARAMETERS:

none

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 21.04.04

(See FEM\_Methods.pro)

-----

PHF::FEMTRUNC

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF::FEMTrunc

PURPOSE:

Correct the coefficients to take into account the  
truncation

CALLING SEQUENCE:

PHF::FEMTrunc, xcof, norml

(See FEM\_Methods.pro)

-----

PHF\_FILE::FEMGET\_XCOF

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF\_FILE::FEMGet\_xcof

PURPOSE:

Extract Legendre coefficients



CALLING SEQUENCE:

PHF\_FILE::FEMGet\_xcof, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

---

PHF\_TTHG::FEMGET\_XCOF

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF\_TTHG::FEMGet\_xcof

PURPOSE:

Extract Legendre coefficients

CALLING SEQUENCE:

PHF\_TTHG::FEMGet\_xcof, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

---

PHF\_RAYL::FEMGET\_XCOF

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF\_RAYL::FEMGet\_xcof

PURPOSE:

Extract Legendre coefficients

CALLING SEQUENCE:

PHF\_RAYL::FEMGet\_xcof, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

---

A\_CONST::FEMGET\_A

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CONST::FEMGet\_a

PURPOSE:

Get a-coefficient for a given wavelength/concentration

CALLING SEQUENCE:

A\_CONST::FEMGet\_a, Ctx, xcof, NXCOF=nxcof

NOTE: this is used in Pure Water definition, in order to reproduce  
a black water condition

(See FEM\_Methods.pro)

-----

A\_PEF::FEMGET\_A

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_PeF::FEMGet\_a

PURPOSE:

Get a-coefficient for a given wavelength  
(for the Pope and Fry model concentration is not used)

CALLING SEQUENCE:

A\_PeF::FEMGet\_a, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

-----

A\_BRI::FEMGET\_A

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_BRI::FEMGet\_a

PURPOSE:

Get a-coefficient for a given wavelength/concentration

CALLING SEQUENCE:

A\_BRI::FEMGet\_a, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

-----

A\_CDOM::FEMGET\_A

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CDOM::FEMGet\_a

PURPOSE:

Get a-coefficient for a given wavelength/concentration

CALLING SEQUENCE:

A\_CDOM::FEMGet\_a, Ctx, xcof, NXCOF=nxcof

NOTE: for this model, the given conc is actually the absorbtion  
coefficient at a given wave lenght (440 um)

(See FEM\_Methods.pro)

-----

A\_CHLA1::FEMGET\_A

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CHLA1::FEMGet\_a

PURPOSE:

Get a-coefficient for a given wavelength/concentration  
This model take into account Chla Absorbtion as from  
Bricaud 95 and adds MIN absorbtion. It is conceived to  
be used with case2 water

CALLING SEQUENCE:

A\_CHLA1::FEMGet\_a, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

-----

A\_CASE1::FEMGET\_A

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CASE1::FEMGet\_a

PURPOSE:

Get a-coefficient for a given wavelength/concentration  
This model take into account Chla Absorbtion as from  
Bricaud 95 and adds CDOM/MIN cohvarying contributions

CALLING SEQUENCE:

A\_CASE1::FEMGet\_a, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

---

B\_CONST::FEMGET\_B

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_CONST::FEMGet\_b

PURPOSE:

Get b-coefficient for a given wavelength/concentration

CALLING SEQUENCE:

B\_CONST::FEMGet\_b, Ctx, xcof, NXCOF=nxcof

NOTE: this is used in Pure Water definition, in order to reproduce  
a black water condition

(See FEM\_Methods.pro)

---

B\_SEB::FEMGET\_B

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_SeB::FEMGet\_b

PURPOSE:

Get b-coefficient for a given wavelength  
For the Smith and Baker concentration is not used

CALLING SEQUENCE:

B\_SeB::FEMGet\_b, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

---

B\_POW::FEMGET\_B

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_POW::FEMGet\_b

PURPOSE:

Get b-coefficient for a given wavelength/concentration

CALLING SEQUENCE:

B\_POW::FEMGet\_b, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

---

B\_MEM::FEMGET\_B

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_MeM::FEMGet\_b

PURPOSE:

Get b-coefficient for a given wavelength/concentration

Compute b coeff. according to Morel/Maritorena model

Conc must be in mg/m3

CALLING SEQUENCE:

B\_MeM::FEMGet\_b, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

---

B\_MIN::FEMGET\_B

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_MIN::FEMGet\_b

PURPOSE:

Get b-coefficient for a given wavelength/concentration

For this model, the given conc is actually the scattering

coefficient at a given wave length (550 um)

CALLING SEQUENCE:

B\_MIN::FEMGet\_b, Ctx, xcof, NXCOF=nxcof

(See FEM\_Methods.pro)

-----

COMP::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength

This method returns the FEM variables for 1 comp.

It already does the phase function truncation correction

All the variables are scaled on the vertical discretisation  
that will be used in the run.

CALLING SEQUENCE:

COMP::FEMGetVar, wave, tau, ssa, xcof

(See FEM\_Methods.pro)

-----

COMP\_PW::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_PW::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength

CALLING SEQUENCE:

COMP\_PW::FEMGetVar, wave, tau, ssa, xcof

PDL:

1. Get a\_w and b\_w from A\_MOD, B\_MOD (SeB or PeF -> read from file)
2. Get PHF from file
3. Compute tau, ssa

(See FEM\_Methods.pro)

---

COMP\_CHLA::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_CHLA::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength

CALLING SEQUENCE:

COMP\_CHLA::FEMGetVar, wave, tau, ssa, xcof

PDL:

Get xcoef and correct

Loop over layers:

```
get Chla concentration from profile
get a(wave,Chla)          -> A_MOD::FEMGet_a
get b(wave,Chla)          -> B_MOD::FEMGet_b
compute tau, ssa
```

Note : keyword NOXCOF not used, as the PHF is always used for  
truncation correction

(See FEM\_Methods.pro)

---

COMP\_CDOM::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_CDOM::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength

CALLING SEQUENCE:

COMP\_CDOM::FEMGetVar, wave, tau, ssa, xcof

PDL:

Loop over layers:

```
get CDOM concentration
get a(wave,Chla)      -> A_MOD::GetAcoeff(wave,Chla)
get b(wave,Chla)      -> only absorbtion !!!
compute tau, ssa
compute/extract xcof  -> 1.0
```

(See FEM\_Methods.pro)

-----

COMP\_MIN::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_MIN::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength

CALLING SEQUENCE:

COMP\_MIN::FEMGetVar, wave, tau, ssa, xcof

PDL:

Loop over layers:

```
get MINE concentration
get a(wave,Chla)      -> only scattering
get b(wave,Chla)      -> A_MOD::GetAcoeff(wave,Chla)
compute tau, ssa
compute/extract xcof
```

Note : keyword NOXCOF not used, as the PHF is always used for  
truncation correction

(See FEM\_Methods.pro)

-----

WAT::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

WAT::FEMGetVar

PURPOSE:



Get tau, ssa, xcof for a given wavelength and vertical profile  
g1, g2 and as are returned only if there is only 1 COMP, and its  
phf is HG

CALLING SEQUENCE:

WAT::FEMGetVar, Ctx, wl, vls, ttau, tssa, txcof, DEBUG=debug, \$  
CTAU=ctau, CSSA=cssa, TIT=tit, NOXCOF=noxcof

(See FEM\_Methods.pro)

-----

COMP\_03::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_03::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength and vertical profile

CALLING SEQUENCE:

COMP\_03::FEMGetVar, Ctx, wl, vls, ttau, tssa, txcof, NOXCOF=noxcof

PDL:

1. Get K\_03 (only for SEAWIFS )
2. Compute the total amount (mol/cm3) for each layer of vls
3. Rescale the optical depth according to gas amount

(See FEM\_Methods.pro)

-----

COMP\_03::FEMGETVAR\_VALID

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_03::FEMGetVar\_valid

PURPOSE:

Get tau, ssa, xcof for a given wavelength and vertical profile  
( only for validation)

CALLING SEQUENCE:

COMP\_03::FEMGetVar\_valid, Ctx, wl, vls, ttau, tssa, txcof, NOXCOF=noxcof

PDL:

1. Get K\_O3 (only for SEAWIFS )
2. Compute the total amount (mol/cm3) for each layer of vls
3. Rescale the optical depth according to gas amount

(See FEM\_Methods.pro)

-----

COMP\_AER::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_AER::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength and vertical profile

CALLING SEQUENCE:

COMP\_AER::FEMGetVar, Ctx, wave, vls, tau, ssa, xcof, NOXCOF=noxcof

PDL:

1. Get aer\_ssa (only for SEAWIFS )
2. Compute tau and ssa as done in FEM
3. Get xcof from phf

(See FEM\_Methods.pro)

-----

COMP\_AER\_OPAC::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_AER\_OPAC::FEMGetVar

PURPOSE:

Load from OPAC file tau, ssa, xcof for a given model, wavelength, vls and relative humidity

CALLING SEQUENCE:

COMP\_AER\_OPAC::FEMGetVar, Ctx, wave, vls, tau, ssa, xcof,  
NOXCOF=noxcof

PDL:

1. Initialisation

2. Read optical properties from ASCII or binary files
3. Rescale tau in boundary layer, according to imposed value at 865 (self.a865)

NOTE on VLS:

1. 'vls' object passed as argument is a FEM-IDL structure containing the overall vertical structure (atm+wat) as vls\_OPAC.bat included in top\_OPAC.bat file. It is used here to get the total number of layers in atmosphere.
2. self.vls is the OPAC vls name, referring to file .dat in /OPAC/../../ofd/ directory. This file is generated by AER\_OPAC.pro->AER\_OPAC\_WRITE\_FEM routine, and contains the description of atm. strato/tropo/boundary layers. It is used, ONLY in case of binary files use (NOT ASCII file), the state the number of layers in strato/tropo/boundary.

Obviously the two description must agree. This is done following the steps below:

- a. Define FEM-IDL vls in FEM environment ( see 1. above)
- b. Use the same levels in opac\_var.bat in OPAC environment.
- c. Generate the file .dat in /OPAC/../../ofd, through AER\_OPAC\_WRITE\_FEM, activating keyword /WRITE\_VLS
- d. Use the name of file at c. in initialising COMP\_AER\_OPAC in system file ( see sys\_OPAC.bat - line : sys\_in.atm.comp2(2,1) = 'OPAC\_01')

NOTE on naming/format: in the current routine, the name and format of the

ASCII/binary files is reproduced (i.e. manually copied)  
from AER\_OPAC.pro->AER\_OPAC\_WRITE\_FEM.

(See FEM\_Methods.pro)

-----

COMP\_AER\_OPAC::FEMGETVARBIN

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_AER\_OPAC::FEMGetVarBin

PURPOSE:

Load from OPAC file tau, ssa, xcof for a given model, wavelength, vls

and relative humidity

CALLING SEQUENCE:

COMP\_AER\_OPAC::FEMGetVar, Ctx, wave, vls, tau, ssa, xcof,  
NOXCOF=noxcof

PDL:

1. Open the OPAC dataset file
2. Load and return tau, ssa, xcof

(See FEM\_Methods.pro)

-----

COMP\_MOL::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_MOL::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength and vertical profile

CALLING SEQUENCE:

COMP\_MOL::FEMGetVar, Ctx, wave, vls, tau, ssa, xcof, P0=p0, NOXCOF=noxcof

PDL:

1. Compute tau according to P0, wave and VLS (as in FEM)
2. Assign xcof (only 2 moments)

(See FEM\_Methods.pro)

-----

COMP\_O2::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_O2::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength and vertical profile

CALLING SEQUENCE:

COMP\_O2::FEMGetVar, Ctx, wave, vls, tau, ssa, xcof, NOXCOF=noxcof

PDL:

1. Get K\_O2 (derived from ~/Seawifs/spectra4.dat)
2. Compute the total amount (mol/cm3) for each layer of vls  
Note that unlike O3, O2 has the same profile for each  
std profile.
3. Rescale the optical depth according to gas amount

Note : can be optimized (save and restore tau,ssa,xcof )

(See FEM\_Methods.pro)

---

ATM::FEMGETVAR

[Previous Routine] [Next Routine] [List of Routines]

NAME:

ATM::FEMGetVar

PURPOSE:

Get tau, ssa, xcof for a given wavelength and vertical profile

ttau[nlyr]	tau for all comps/ every lyrs (tot)
tssa[nlyr]	ssa for all comps/ every lyrs (tot)
/ctau[ncomp,nlyr]	tau for every comps/lyrs (comp)
/cssa[ncomp,nlyr]	ssa for every comps/lyrs (comp)

(See FEM\_Methods.pro)

---

SYSTEM::FEM\_INPUT

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SYSTEM::FEM\_Input

PURPOSE:

Extract from a 'system' object all the information needed for a  
FEM run and copy to str\_in1, str\_in2, str\_mom

CALLING SEQUENCE:

SYSTEM::FEM\_Input, Ctx, vls, str\_in1, str\_in2, str\_mom, BIN=bin, \$  
DBG\_ATM=dbg\_atm, DBG\_WAT=dbg\_wat, TIT=tit

(See FEM\_Methods.pro)

---

SYSTEM::FEM\_OPTPRO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SYSTEM::FEM\_OPTPRO

PURPOSE:

Compute for atmosphere and water the following:

atot_tau [nlyr]	atm. tau for all comps/ every lyrs
atot_ssa [nlyr]	atm. ssa for all comps/ every lyrs
acmp_tau [ncomp,nlyr]	atm. tau for every comps/lyrs
acmp_ssa [ncomp,nlyr]	atm. ssa for every comps/lyrs
wtot_tau [nlyr]	wat. tau for all comps/ every lyrs
wtot_ssa [nlyr]	wat. ssa for all comps/ every lyrs
wcmp_tau [ncomp,nlyr]	wat. tau for every comps/lyrs
wcmp_ssa [ncomp,nlyr]	wat. ssa for every comps/lyrs
wtot_a [ncomp,nlyr]	wat. a coeff for all comps/ every lyrs
wtot_b [ncomp,nlyr]	wat. b coeff for all comps/ every lyrs
wcmp_a [ncomp,nlyr]	wat. a coeff for every comps/lyrs
wcmp_b [ncomp,nlyr]	wat. b coeff for every comps/lyrs

KEYWORDS:

NOATM: do not compute for atmosphere  
NOWAT: do not compute for water  
NOCOMPS: do not compute tau/ssa for components  
NOXCOF: do not compute xcoeff (time/memory consuming)

Note : system is defined for 1 wavelength !!!

(See FEM\_Methods.pro)

---

SYSTEM::TEST\_OPAC

[Previous Routine] [List of Routines]

NAME:

SYSTEM::TEST\_OPAC

PURPOSE:

Load and check/compare OPAC dataset files, in ASCII and BINARY format

CALLING SEQUENCE:

SYSTEM::FEM\_Input, Ctx

(See FEM\_Methods.pro)

-----

## 4.3 List of Hydrolight-IDL routines

Hydrolight-IDL routine list and description

This page was created by the IDL library routine `mk_html_help`. For more information on this routine, refer to the IDL Online Help Navigator or type:

```
? mk_html_help
```

at the IDL command line prompt.

Last modified: Mon Jan 31 17:22:07 2005.

---

### List of Routines

- \* HYD\_AC\_DEFINE\_STRUCT
- \* HYD\_AC\_INIT\_STR\_IN
- \* HYD\_AC\_ALLOC\_STR\_IN
- \* HYD\_AC\_INIT\_STR\_IN\_1
- \* HYD\_AC\_WRITE\_INPUT1
- \* HYD\_AC\_READ\_DIGITAL
- \* HYD\_DD\_WAT\_COEFF
- \* HYD\_DD\_RADIANCE
- \* HYD\_DD\_IRRADIANCE
- \* HYD\_PP\_RUN
- \* HYD\_PP\_READ
- \* HYD\_TL\_GET\_CONTEXT
- \* HYD\_DEFINE\_CLASSES
- \* SKY::HYDIN1
- \* SKY1::HYDGETINFO
- \* SKY2::HYDGETINFO
- \* SKY3::HYDGETINFO
- \* VPROFILE::HYDGETINFO
- \* VPROFILE::HYDWRITEFILE
- \* UNIFORM::HYDGETINFO
- \* GAUSSIAN::HYDGETINFO
- \* SERIE::HYDGETINFO
- \* ISRC::HYDGETINFO
- \* PHF\_RATIO::INITIAL
- \* PHF\_HYSCA::INITIAL
- \* PHF\_CDOM::INITIAL
- \* PHF\_FILE::HYDGETINFO
- \* PHF\_RATIO::HYDGETINFO
- \* PHF\_HYSCA::HYDGETINFO
- \* PHF\_CDOM::HYDGETINFO
- \* A\_CONST::HYDGETINFO



- \* A\_USR::HYDGETINFO
- \* A\_PEF::HYDGETINFO
- \* A\_SEB::HYDGETINFO
- \* A\_PSM::HYDGETINFO
- \* A\_EXP::HYDGETINFO
- \* A\_CDOM::HYDGETINFO
- \* B\_CONST::HYDGETINFO
- \* B\_USR::HYDGETINFO
- \* B\_SEB::HYDGETINFO
- \* B\_POW::HYDGETINFO
- \* B\_GAM::HYDGETINFO
- \* B\_GEM::HYDGETINFO
- \* B\_CDOM::HYDGETINFO
- \* COMP\_CONST::HYDGETINFO
- \* COMP\_PW::HYDGETINFO
- \* COMP\_CHLA::HYDGETINFO
- \* COMP\_CDOM::HYDGETINFO
- \* COMP\_MIN::HYDGETINFO
- \* WAT::HYDIN1
- \* WAT\_CONST::HYDIN1
- \* WAT\_CASE1::HYDIN1
- \* WAT\_CASE2::HYDIN1
- \* WAVE::HYDGETINFO
- \* WAVE::HYDIN1
- \* OLS::HYDGETINFO
- \* OLS::HYDIN1
- \* BOTTOM::HYDGETINFO
- \* BOTTOM::HYDIN1
- \* ATM::HYDGETINFO
- \* ATM::HYDIN1
- \* SYSTEM::HYD\_INPUT

-----

## Routine Descriptions

### HYD\_AC\_DEFINE\_STRUCT

[Next Routine] [List of Routines]

#### NAME:

HYD\_AC\_DEFINE\_STRUCT

#### PURPOSE:

Define structures fo I/O

#### CATEGORY:

HYD Access

CALLING SEQUENCE:

Result = HYD\_AC\_DEFINE\_STRUCT()

INPUTS:

Ctx : global context

KEYWORD PARAMETERS:

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 10.05.04

(See HYD\_Access.pro)

-----

HYD\_AC\_INIT\_STR\_IN

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_AC\_INIT\_STR\_IN

PURPOSE:

Initialise structure STR\_INPUT1

CATEGORY:

HYD Access

CALLING SEQUENCE:

Result = HYD\_AC\_INIT\_STR\_IN(str\_in)

INPUTS:

str\_in1 : structure for input file

KEYWORD PARAMETERS:

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 17.05.04

(See HYD\_Access.pro)

-----

HYD\_AC\_ALLOC\_STR\_IN

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_AC\_ALLOC\_STR\_IN

PURPOSE:

Allocate the array depending on variable dimensions

CATEGORY:

HYD Access

CALLING SEQUENCE:

Result = HYD\_AC\_ALLOC\_STR\_IN(str\_in,ncomp,nconc)

INPUTS:

ncomp : number of components  
nconc : number of concentrations  
str\_in1 : structure for input file

KEYWORD PARAMETERS:

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 17.05.04

(See HYD\_Access.pro)

-----

HYD\_AC\_INIT\_STR\_IN\_1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_AC\_INIT\_STR\_IN\_1

PURPOSE:

Initialise structure STR\_INPUT1 for the UG example 1

CATEGORY:

HYD Access

CALLING SEQUENCE:

Result = HYD\_AC\_INIT\_STR\_IN\_1(str\_in)

INPUTS:

str\_in1 : structure for input file

KEYWORD PARAMETERS:

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 13.05.04

(See HYD\_Access.pro)

-----

HYD\_AC\_WRITE\_INPUT1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_AC\_WRITE\_INPUT1

PURPOSE:

Write the HYD input file 1

CATEGORY:

HYD Access

CALLING SEQUENCE:

Result = HYD\_AC\_WRITE\_INPUT1(Ctx,Str\_in1)

INPUTS:

Ctx : global context  
Str\_in1 : structure containing data to be written

KEYWORD PARAMETERS:

dir : input direct; if not defined is taken from Ctx  
file : input name 1 as define for HYD in iolist.txt  
if not defined is taken from Ctx

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 13.05.04

(See HYD\_Access.pro)

-----

HYD\_AC\_READ\_DIGITAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_AC\_READ\_DIGITAL

PURPOSE:

Read the 'digital' output file

CATEGORY:

HYD Access

CALLING SEQUENCE:

Result = HYD\_AC\_READ\_DIGITAL(Ctx,str\_out)

INPUTS:

Ctx : global context  
Str\_out : structure containing read data {STR\_OUTDIG}

KEYWORD PARAMETERS:

dir : output direct; if not defined is taken from Ctx  
file : output file name. If not defined is build based on Ctx.RID

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 21.05.04

(See HYD\_Access.pro)

-----

HYD\_DD\_WAT\_COEFF

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_DD\_WAT\_COEFF

PURPOSE:

Display an wat coefficient (a,b,c,kd..)

CATEGORY:

HYD Display

CALLING SEQUENCE:

HYD\_DD\_WAT\_COEFF, str\_out, ..

INPUTS:

str\_out : structure containing Hydrolight results

KEYWORD PARAMETERS:

TYPE	: coefficient type ('a','b',...) - default is 'a'
SURF	: 3D plot
COMP	: display coeff. for each component (incomp. with SURF) otherwise only total value
IWAVE	: used to select a wavelength, when COMP>=1
OPTIC	: plot vs. optical depth
EPS	: print to Encapsulated Postscript file
PS	: print to Postscript file

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 25.05.04

(See HYD\_Display.pro)

-----

## HYD\_DD\_RADIANCE

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

### NAME:

HYD\_DD\_RADIANCE

### PURPOSE:

Display a radiance field as function of viewing dir and wave

### CATEGORY:

HYD Display

### CALLING SEQUENCE:

HYD\_DD\_RADIANCE, str\_out, ..

### INPUTS:

str\_out : structure containing Hydrolight results

### KEYWORD PARAMETERS:

TYPE : radiance type (0 -> diffuse, 1 -> total)

REFL : include surf. reflectance, only if depth=0 (air) Def. is 0.

DEPTH : index for the depth (0 -> in air ; 1 -> first water layer (default))

PHI : index for the azimuth (default is principal plane)

LOG : set radiance log axis

SURF : 3D plot

EPS : print to Encapsulated Postscript file

PS : print to Postscript file

### OUTPUTS:

none

### MODIFICATION HISTORY:

Written by: Marco Clerici, 24.05.04

(See HYD\_Display.pro)

-----

## HYD\_DD\_IRRADIAN

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

HYD\_DD\_IRRADIANT

PURPOSE:

Display an irradiance field as function of depth and wave

CATEGORY:

HYD Display

CALLING SEQUENCE:

HYD\_DD\_IRRADIANT, str\_out, ..

INPUTS:

str\_out : structure containing Hydrolight results

KEYWORD PARAMETERS:

TYPE : irradiance type ('Ed', 'Eu', 'Eou', 'Eod')  
LOG : use logarithmic axis  
SURF : 3D plot  
OPTIC : plot vs. optical depth  
EPS : print to Encapsulated Postscript file  
PS : print to Postscript file

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 26.05.04

(See HYD\_Display.pro)

-----

HYD\_PP\_RUN

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_PP\_RUN

PURPOSE:

Write the input files and make the HYD run

CATEGORY:



HYD Access

CALLING SEQUENCE:

Result = HYD\_PP\_RUN(Ctx,...)

INPUTS:

Ctx : global context  
Str\_in1 : structure containing all the data for a run

KEYWORD PARAMETERS:

RID : run ID - if set, the output files name is modified, e.g.  
output1name\_RAD.TXT -> output1name\_'RID'\_RAD.TXT  
IN\_DIR : input directory (if not set the one from Ctx is used)  
SYSTEM : system object to be provided if Str\_in1 is not set.  
SKYTYPE : passed to HYD\_Input

OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

MODIFICATION HISTORY:

Written by: Marco Clerici, 24.05.04

(See HYD\_Process.pro)

-----

HYD\_PP\_READ

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_PP\_READ

PURPOSE:

Read the results of a HYD run

CATEGORY:

HYD Access

CALLING SEQUENCE:

Result = HYD\_PP\_READ(Ctx,...)

#### INPUTS:

Ctx : global context  
Str\_out1 : Output1 structure (used to read 'report' file)  
Str\_out2 : Output2 structure (used to read radiance from binary file)  
Str\_out4 : Output4 structure (used to read fluxes from binary file)

#### KEYWORD PARAMETERS:

RID : run IF - if set, the output files name is modified, e.g.  
output1name\_RAD.TXT -> output1name\_'RID'\_RAD.TXT  
DIR : output directory ( if not set the one from Ctx is used)

#### OUTPUTS:

RETURN : 0 - File written OK  
>0 - Error in file writing

NOTE: output structures are generated by the calling routines. If an output  
does not have to be read, put 0 as argument instead of the structure

#### MODIFICATION HISTORY:

Written by: Marco Clerici, 22.04.04

(See HYD\_Process.pro)

-----

#### HYD\_TL\_GET\_CONTEXT

[Previous Routine] [Next Routine] [List of Routines]

#### NAME:

HYD\_TL\_GET\_CONTEXT

#### PURPOSE:

This function creates the general context and initialises it.

The general context hosts the User Defined set-up Parameters and the  
Global Constants used by HYD.

The User Defined set-up Parameters are initialised with default  
values that can be overwritten by the values defined in the Configu-  
ration File.

The Global Constants are hard-coded and SHOULD NOT be modified by the user.

CATEGORY:

HYD Tool

CALLING SEQUENCE:

Result = HYD\_TL\_GET\_CONTEXT(Ctx, [,DIR=dir] [,/PRINT])

INPUTS:

none

KEYWORD PARAMETERS:

DIR : Directory containing Configuration File  
PRINT : Print User Defined Parameters after reading

OUTPUTS:

ctx : General HYD context  
RETURN : 0 - Success  
          >0 - Failure

MODIFICATION HISTORY:

Written by: Marco Clerici, 08.01.03

(See HYD\_Tool.pro)

-----

HYD\_DEFINE\_CLASSES

[Previous Routine] [Next Routine] [List of Routines]

NAME:

HYD\_DEFINE\_CLASSES

PURPOSE:

Define HYD specific classes, not defined at RTM (system) level:  
SKY : define sky behaviour  
PHF specific classes

CATEGORY:

HYD Methods

CALLING SEQUENCE:

HYD\_DEFINE\_CLASSES

INPUTS:

none

KEYWORD PARAMETERS:

none

OUTPUTS:

none

MODIFICATION HISTORY:

Written by: Marco Clerici, 21.05.04

(See HYD\_Methods.pro)

-----

SKY::HYDIN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SKY::HydIN1

PURPOSE:

Get SKY properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file.

CALLING SEQUENCE:

SKY::HydIN1, str\_in1

(See HYD\_Methods.pro)

-----

SKY1::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SKY1::HydGetInfo

PURPOSE:

Convert SKY1 properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

SKY1::HydGetInfo, flagsky, nsky, argum

(See HYD\_Methods.pro)

---

SKY2::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SKY2::HydGetInfo

PURPOSE:

Convert SKY2 properties in Hydrolight input flags/  
parameters.

argum is : sunzen, sunphi, cloud

CALLING SEQUENCE:

SKY2::HydGetInfo, flagsky, nsky, argum

(See HYD\_Methods.pro)

---

SKY3::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SKY3::HydGetInfo

PURPOSE:

Convert SKY2 properties in Hydrolight input flags/  
parameters.

argum is : jday, rlat, rlon, hour, sunphi, cloud

CALLING SEQUENCE:

SKY3::HydGetInfo, flagsky, nsky, argum

(See HYD\_Methods.pro)

---

VPROFILE::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

VPROFILE::HydGetInfo

PURPOSE:

Convert VPROFILE properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

VPROFILE::HydGetInfo, itype, file, routine

(See HYD\_Methods.pro)

---

VPROFILE::HYDWRITEFILE

[Previous Routine] [Next Routine] [List of Routines]

NAME:

VPROFILE::HydWriteFile

PURPOSE:

Write Vertical Profile in Hydrolight input format

CALLING SEQUENCE:

VPROFILE::HydWriteFile, n, step

(See HYD\_Methods.pro)

---

UNIFORM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

UNIFORM::HydGetInfo

PURPOSE:

Convert UNIFORM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

UNIFORM::HydGetInfo, itype, file, routine

(See HYD\_Methods.pro)

---

GAUSSIAN::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

GAUSSIAN::HydGetInfo

PURPOSE:

Convert GAUSSIAN properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

GAUSSIAN::HydGetInfo, itype, file, routine

(See HYD\_Methods.pro)

---

SERIE::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

SERIE::HydGetInfo

PURPOSE:

Convert SERIE properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

SERIE::HydGetInfo, itype, file, routine

(See HYD\_Methods.pro)

---

ISRC::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

ISRC::HydGetInfo

PURPOSE:

Convert ISRC properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

ISRC::HydGetInfo, itype, file, routine

(See HYD\_Methods.pro)

---

PHF\_RATIO::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF\_RATIO::Initial

PURPOSE:

Initialise a PHF\_RATIO object (only for Hydrolight)  
A PHF is build to match the given bb/b ratio

CALLING SEQUENCE:

PHF\_RATIO::Initial [,ID=id] [RATIO=ratio]

INPUT:

none

KEYWORD:

ID: phase function identifier  
RATIO: value of the bb/b ratio to match

(See HYD\_Methods.pro)

---

PHF\_HYSCA::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF\_HYSCA::Initial

PURPOSE:

Initialise a PHF\_RATIO PHF\_HYSCA (only for Hydrolight)  
A HydroScat phase function is chosen according to the  
given name

CALLING SEQUENCE:

PHF\_HYSCA::Initial [,ID=id] [,NAME=name] [,DELTA=delta]

INPUT:

none

KEYWORD:

ID: phase function identifier



NAME: HydroScat file name  
DELTA: bb/b ratio tolerance

(See HYD\_Methods.pro)

-----

PHF\_CDOM::INITIAL

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF\_CDOM::Initial

PURPOSE:

Initialise a PHF\_CDOM object (only for Hydrolight)  
A PHF is selected according to file name

CALLING SEQUENCE:

PHF\_CDOM::Initial [,ID=id] [,FILE=file]

INPUT:

none

KEYWORD:

ID: phase function identifier  
FILE: PHF file name

(See HYD\_Methods.pro)

-----

PHF\_FILE::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

PHF\_FILE::HydGetInfo

PURPOSE:

Convert PHF\_FILE properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

PHF\_FILE::HydGetInfo, filename, ibbopt, bbfrac, delta

(See HYD\_Methods.pro)

---

PHF\_RATIO::HYDGETINFO

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

PHF\_RATIO::HydGetInfo

PURPOSE:

Convert PHF\_RATIO properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

PHF\_RATIO::HydGetInfo, filename, ibbopt, bbfrac, delta

(See HYD\_Methods.pro)

---

PHF\_HYSCA::HYDGETINFO

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

PHF\_HYSCA::HydGetInfo

PURPOSE:

Convert PHF\_HYSCA properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

PHF\_HYSCA::HydGetInfo, filename, ibbopt, bbfrac, delta

(See HYD\_Methods.pro)

---

PHF\_CDOM::HYDGETINFO

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

PHF\_CDOM::HydGetInfo

PURPOSE:

Convert PHF\_CDOM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

PHF\_CDOM::HydGetInfo, filename, ibbopt, bbfrac, delta

(See HYD\_Methods.pro)

-----

A\_CONST::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CONST::HydGetInfo

PURPOSE:

Convert A\_CONST properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

A\_CONST::HydGetInfo, filename, iastropt, astarRef, astar0, asgam

(See HYD\_Methods.pro)

-----

A\_USR::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_USR::HydGetInfo

PURPOSE:

Convert A\_USR properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

A\_USR::HydGetInfo, filename, iastropt, astarRef, astar0, asgam

(See HYD\_Methods.pro)

-----

A\_PEF::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_PeF::HydGetInfo

PURPOSE:

Convert A\_PeF properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

A\_PeF::HydGetInfo, filename, iastropt, astarRef, astar0, asgam

(See HYD\_Methods.pro)

---

A\_SEB::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_SeB::HydGetInfo

PURPOSE:

Convert A\_SeB properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

A\_SeB::HydGetInfo, filename, iastropt, astarRef, astar0, asgam

(See HYD\_Methods.pro)

---

A\_PSM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_PSM::HydGetInfo

PURPOSE:

Convert A\_PSM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

A\_PSM::HydGetInfo, filename, iastropt, astarRef, astar0, asgam

(See HYD\_Methods.pro)

---

A\_EXP::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_EXP::HydGetInfo

PURPOSE:

Convert A\_EXP properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

A\_EXP::HydGetInfo, filename, iastropt, astarRef, astar0, asgam

(See HYD\_Methods.pro)

---

A\_CDOM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

A\_CDOM::HydGetInfo

PURPOSE:

Convert A\_CDOM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

A\_CDOM::HydGetInfo, filename, iastropt, astarRef, astar0, asgam

(See HYD\_Methods.pro)

---

B\_CONST::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_CONST::HydGetInfo

PURPOSE:

Convert B\_CONST properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

B\_CONST::HydGetInfo, filename, ibstropt, bstarRef, bstar0, coef

(See HYD\_Methods.pro)

---

B\_USR::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_USR::HydGetInfo

PURPOSE:

Convert B\_USR properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

B\_USR::HydGetInfo, filename, ibstropt, bstarRef, bstar0, coef

(See HYD\_Methods.pro)

---

B\_SEB::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_SeB::HydGetInfo

PURPOSE:

Convert B\_SeB properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

B\_SeB::HydGetInfo, filename, ibstropt, bstarRef, bstar0, coef

(See HYD\_Methods.pro)

---

B\_POW::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_POW::HydGetInfo

PURPOSE:

Convert B\_POW properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

B\_POW::HydGetInfo, filename, ibstropt, bstarRef, bstar0, coef

(See HYD\_Methods.pro)

-----

B\_GAM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_GAM::HydGetInfo

PURPOSE:

Convert B\_GAM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

B\_GAM::HydGetInfo, filename, ibstropt, bstarRef, bstar0, coef

(See HYD\_Methods.pro)

-----

B\_GEM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_GeM::HydGetInfo

PURPOSE:

Convert B\_GeM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

B\_GeM::HydGetInfo, filename, ibstropt, bstarRef, bstar0, coef

(See HYD\_Methods.pro)

---

B\_CDOM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

B\_CDOM::HydGetInfo

PURPOSE:

Convert B\_CDOM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

B\_CDOM::HydGetInfo, filename, ibstropt, bstarRef, bstar0, coef

(See HYD\_Methods.pro)

---

COMP\_CONST::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_CONST::HydGetInfo

PURPOSE:

Convert COMP\_CONST properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

COMP\_CONST::HydGetInfo, itype, conc\_file

(See HYD\_Methods.pro)

---

COMP\_PW::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_PW::HydGetInfo

PURPOSE:

Convert COMP\_PW properties in Hydrolight input flags/  
parameters.



CALLING SEQUENCE:

COMP\_PW::HydGetInfo, itype, conc\_file

(See HYD\_Methods.pro)

---

COMP\_CHLA::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_CHLA::HydGetInfo

PURPOSE:

Convert COMP\_CHLA properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

COMP\_CHLA::HydGetInfo, itype, conc\_file

(See HYD\_Methods.pro)

---

COMP\_CDOM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_CDOM::HydGetInfo

PURPOSE:

Convert COMP\_CDOM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

COMP\_CDOM::HydGetInfo, itype, conc\_file

(See HYD\_Methods.pro)

---

COMP\_MIN::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

COMP\_MIN::HydGetInfo

PURPOSE:

Convert COMP\_MIN properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

COMP\_MIN::HydGetInfo, itype, conc\_file

(See HYD\_Methods.pro)

-----

WAT::HYDIN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

WAT::HydIN1

PURPOSE:

Get WAT properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file.

CALLING SEQUENCE:

WAT::HydIN1, str\_in1

(See HYD\_Methods.pro)

-----

WAT\_CONST::HYDIN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

WAT\_CONST::HydIN1

PURPOSE:

Get WAT\_CONST properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file.

CALLING SEQUENCE:

WAT\_CONST::HydIN1, str\_in1

(See HYD\_Methods.pro)

---

WAT\_CASE1::HYDIN1

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

WAT\_CASE1::HydIN1

PURPOSE:

Get WAT\_CASE1 properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file, just calling WAT::HydIN1()

CALLING SEQUENCE:

WAT\_CONST::HydIN1, str\_in1

(See HYD\_Methods.pro)

---

WAT\_CASE2::HYDIN1

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

WAT\_CASE2::HydIN1

PURPOSE:

Get WAT\_CONST properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file, just calling WAT::HydIN1()

CALLING SEQUENCE:

WAT\_CASE2::HydIN1, str\_in1

(See HYD\_Methods.pro)

---

WAVE::HYDGETINFO

[\[Previous Routine\]](#) [\[Next Routine\]](#) [\[List of Routines\]](#)

NAME:

WAVE::HydGetInfo

PURPOSE:

Convert WAVE properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

WAVE::HydGetInfo, nwave, wave

(See HYD\_Methods.pro)

---

WAVE::HYDIN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

WAVE::HydIN1

PURPOSE:

Get WAVE properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file.

CALLING SEQUENCE:

WAVE::HydIN1, str\_in1

(See HYD\_Methods.pro)

---

OLS::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

OLS::HydGetInfo

PURPOSE:

Convert OLS properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

OLS::HydGetInfo, wat, nznom, znom

(See HYD\_Methods.pro)

---

OLS::HYDIN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

OLS::HydIN1

PURPOSE:

Get OLS properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file.

CALLING SEQUENCE:

OLS::HydIN1, str\_in1

(See HYD\_Methods.pro)

-----

BOTTOM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

BOTTOM::HydGetInfo

PURPOSE:

Convert BOTTOM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

BOTTOM::HydGetInfo, ibotm, rflbot, file

(See HYD\_Methods.pro)

-----

BOTTOM::HYDIN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

BOTTOM::HydIN1

PURPOSE:

Get BOTTOM properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file.

CALLING SEQUENCE:

BOTTOM::HydIN1, str\_in1

(See HYD\_Methods.pro)

---

ATM::HYDGETINFO

[Previous Routine] [Next Routine] [List of Routines]

NAME:

ATM::HydGetInfo

PURPOSE:

Convert ATM properties in Hydrolight input flags/  
parameters.

CALLING SEQUENCE:

ATM::HydGetInfo, wspeed

(See HYD\_Methods.pro)

---

ATM::HYDIN1

[Previous Routine] [Next Routine] [List of Routines]

NAME:

ATM::HydIN1

PURPOSE:

Get ATM properties and assign to structure to write  
Hydrolight 4.1 ASCII Input file.

CALLING SEQUENCE:

ATM::HydIN1, str\_in1

(See HYD\_Methods.pro)

---

SYSTEM::HYD\_INPUT

[Previous Routine] [List of Routines]

NAME:

SYSTEM::HYD\_Input

PURPOSE:

Drive the whole preparation of Hydrolight input file, calling  
hierachically HydIN1() method for all System objects

Handle also the sky type: different 'sky' method can be used in Hydrolight, which correspond to the same 'system'

If SKYTYPE not set, sky must have already initialised str\_in (as in test\_FEM.bat)

CALLING SEQUENCE:

SYSTEM::HYD\_Input, str\_in1, SKYTYPE= skytype

(See HYD\_Methods.pro)

---

# References

- Bricaud, A., M. Babin, A. Morel, and H. Claustre (1995). Variability in the chlorophyll-specific absorption coefficients of natural phytoplankton: analysis and parameterization. *J. Geophys. Res.*, *100*, 13321–13332.
- Bricaud, A., A. Morel, M. Babin, K. Allali, and H. Claustre (1998). Variations of light absorption by suspended particles with chlorophyll *a* concentration in oceanic (case 1) waters: analysis and implications for bio-optical models. *J. Geophys. Res.*, *103*, 31033–31044.
- Bricaud, A., A. Morel, and L. Prieur (1981). Absorption by dissolved organic matter of the sea (yellow substance) in the UV and visible domains. *Limnol. Oceanogr.*, *26*, 43–53.
- Bulgarelli, B., V. Kisselev, and L. Roberti (1999). Radiative transfer in the atmosphere-ocean system: The finite-element method. *Appl. Opt.*, *38*, 1530–1542.
- Gordon, H. and A. Morel (1983). Remote assessment of ocean color for interpretation of satellite visible imagery, a review. *Lecture Notes on Coastal and Estuarine Studies* *4*, 114.
- Hansen, J. E. and L. Travis (1974). Light scattering in planetary atmospheres. *Space Science Reviews*, *16*, 527–610.
- Heney, L. and J. Greenstein (1941). Diffuse radiation in the galaxy. *Astrophys. J.*, *93*, 70–83.
- Hess, M., P. Koepke, and I. Schult (1998). Optical properties of aerosol and clouds: The software package opac. *Bull. Amer. Meteor. Soc.*, *79*, 831–844.
- Mobley, C. (1994). *Light and Water*. San Diego: Academic Press, Inc.
- Mobley, C. and L. Sundman (2000a). *Hydrolight 4.1 Technical Documentation* (First Printing ed.). Redmond.
- Mobley, C. and L. Sundman (2000b). *Hydrolight 4.1 User's Guide* (First Printing ed.). Redmond.
- Morel, A. and S. Maritorena (2001). Bio-optical properties of oceanic waters: A reappraisal. *J. Geophys. Res.*, *106*, 7163–7180.
- Nicolet, M. (1981). The solar spectral irradiance and its action in the atmospheric photodissociation processes. *Planet. Space Sci.* *29*, 951–974.
- Pope, R. and E. Fry (1997). Absorption spectrum (380–700 nm) of pure water. II. Integrating cavity measurements. *Appl. Opt.*, *36*, 8710–8723.
- Prieur, L. and S. Sathyendranath (1981). An optical classification of coastal and oceanic waters based on the specific absorption of phytoplankton pigments, dissolved organic matter, and other particulate materials. *Limnol. Oceanogr.*, *26*, 671–689.
- Smith, R. and K. Baker (1981). Optical properties of the clearest natural waters. *Appl. Opt.*, *20*, 177–184.
- Thomas, G. and K. Stamnes (1999). *Radiative Transfer in the Atmosphere and Ocean*. Cambridge: Cambridge University Press.



Wiscombe, W. (1977). The delta-M method: Rapid yet accurate radiative flux calculations for strongly asymmetric phase functions. *J. Atmos. Sci.*, *34*, 1408–1422.